بسم الله الرحمن الرحيم
قال تعالى "وقل رب زدني علما"

ملاحظة: لغة لتربو باسكال لا تميز بين الأحرف الكبيرة و الصغيرة في تسمية المتحولات وأسماء التوابع و الإجرائيات.
(هناك فرق بين لغة تربو باسكال وباسكال القياسية في كيفية كتابة و قراءة المعلومات من الملفات محددة النوع و قد جاءت لغة تربو باسكال لتبسيط و تسهيل التعامل مع الملفات محددة النوع)

الملفات FILES:

قد نحتاج في بعض الأحيان إلى إدخالات أو إخراجات تملك صفة الاستمرار أي أنها لا تتغير بمجرد تنفيذ البرنامج (قاعدة بيانات) ومن أجل ذلك نستخدم الملفات في لغة باسكال إذ إن الملف عبارة عن مكان يبحث فيه البرنامج عما يحتاجه وفق أسلوب منطقي logic manner وبشكل مستقل عن طريقة توضع هذه الملفات ضمن القرص disk .حيث تتوضع محتويات الملف تحت نظام التشغيل DOS في مواقع فيزيائية متنوعة على القرص وليس ضمن مواقع متتالية كما يبدو لأول وهلة .

يُقرأ الملف عنصراً عنصراً فبعد أن يقرأ عنصر ما يفترض البرنامج أنه يستطيع قراءة العنصر التالي و هكذا دواليك مادام هناك عناصر متبقية لم تقرأ من الملف.

تجري عملية القراءة باستخدام المشغلات driver التي يزودنا بها نظام DOS (تدعى أحياناً بالسواقات) وبمساعدة بيئة تربو باسكال تبعدنا هذه الوسائل عن التعامل مع الملفات وفق الصيغة الفيزيائية المخزنة على القرص وتمكن البرنامج من جلب عناصر الملف بدون تعقيد.

يرتبط اسم الملف بمحتوياته أي أن اسم الملف يشير إلى هذا الملف عند سطر أوامر DOS. فعلى سبيل المثال :من أجل البحث عن ملف اسمه DiskFile.NAM في دليل ما يمكننا كتابة الأمر التالي عند محث نظام DOS:

dir diskfile.nam

فإذا وجد الملف في الدليل الحالي فإن نظام التشغيل DOS سوف يزودنا بمعلومات عن حجم هذا الملف و تاريخ إنشائه أو تاريخ تعديل طرأ عليه و هكذا.........دواليك.

سوف أشرح كيفية استخدام الملفات في برامج تربو باسكال أي سوف نتعلم كيف ننشئ ونفتح ونغلق الملفات وسوف نتعلم سوياً كيف نقرأ أو نكتب المعلومات في الملفات.

الملفات النصية text files:

الملف في الواقع عبارة عن مفهوم منطقي لا يدل على نوع المعلومات الموجودة ضمن هذا الملف لأننا نستطيع تخزين أنواع متعددة من المعلومات ضمن الملفات قد تكون سلاسل رمزية strings أو قيم عددية لذلك يطلق على الملف النصي text file بملف ASCII (ASCII file) حيث يحتوي هذا الملف على معلومات ممثلة بشفرة ASCII ويمكننا قراءة الملفات النصية بواسطة أي محرر نصوص text editor أو باستخدام الأمر type في النظام DOS .

يوجد نوع آخر من الملفات يسمى بالملفات الثنائية binary files تخزن معلوماتها بالشفرة الثنائية وليست وفق ASCII و هذه الملفات غير مفهومة لمعظم محررات النصوص.

ولدينا التعليمات readln,read,writeln,write تستخدم في عمليات الإدخال والإخراج التي تطبق على الملفات النصية .يوجد نوع معطيات مسبق التعريف بلغة باسكال هو نوع المعطيات ملف نصي text type يستخدم هذا النوع لتمثيل الملفات النصية . والسطر التالي يرينا كيفية التصريح عن متحول ينتمي إلى نوع المعطيات هذا:

Var fileval:text;

العمليات على الملفات Actions On Files:

لا يوجد الكثير من العمليات التي يمكن تطبيقها على الملفات سوى عملية فتح وإغلاق الملفات فمن أجل فتح openأو إنشاء Create ملف في لغة تربو باسكال علينا أولا ربط اسم متحول هذا الملف (المصرح عنه ضمن البرنامج و غير الموجود (المعرف) خارج حدود البرنامج) باسم الملف الذي يعرفه به نظام التشغيل.

تُمكن عملية الربط هذه الملف البرنامج من الوصول إلى الملف الموجود على القرص ويمكن مشغلات نظام التشغيل من تسلم زمام عملية التعامل مع الملف الموجود على القرص . والخطوة التالية هي فتح الملف للاستخدام ضمن الملف إذ توجد إجراءات مسبقة التعريف للقيام بهذه العملية و القيام بالعملية المعاكسة و هي إغلاق الملف close file و بالتالي تخزين محتوياته على القرص.

ربط البرنامج بالملف الموجود على القرص Associating a program and a Disk File:

يعمل الإجراء Assign على ربط اسم بمتحول الملف فمثلا تربط العبارة التالي الاسم ZQZQ.QZQ بمتحول الملف النصي samplefile:

ASSIGN(sample,'zQzQ.QzQ');

تفترض العبارة السابقة أننا صرحنا عن المتحول السابق على أنه من النوع ملف نصي text files وكما هو موضح في العبارة التالية:

Var samplefile:text;

يجب الانتباه إلى أن الإجراء ASSIGN يجب أن يكون ضمن مجال التأثير scope للتصريح عن المتحول samplefile وأن اسم الملف قد خصص على شكل سلسلة رمزية في حين أن متحول الملف قد مُثل بواسطة مميز ينتمي إلى نوع المعطيات text.

متحول الملف file variable واسم الملف file name يستخدمان ضمن سياقات مختلفة حيث يستخدم متحول الملف عندما نريد الإشارة إلى الملف ضمن البرنامج. فمثلاً :تكتب العبارة التالية العدد (395.7) على الملف samplefile:

Writeln(samplefile,395.7);

أما اسم الملف الحقيقي على القرص فلا يستخدم ضمن البرنامج و لكن يستخدم خارج البرنامج فعلى سبيل المثال :ضمن نظام DOS إذا كان لدينا الملف ZQZQ.QZQ في الدليل الحالي يمكننا كتابة الأمر التالي من أوامر DOS:

type zqzq.qzq

سوف تُظهر التعليمة السابقة محتويات الملف المسمى zqzq.qzq على الشاشة حيث أن النظام DOS يفهم الاسم ZQZQ.qzq على أنه يشير إلى ملف مخزن على القرص في حين لا يعرف مميز متحول الملف samplefile المستخدم ضمن البرنامج لأن هذا المتحول سوف يختفي عندما يتوقف تنفيذ البرنامج.

وباستخدام عملية ربط متحول الملف باسم الملف بواسطة استدعاء الإجراء assign استطعنا الإشارة إلى المتحول ذي النوع text خارج حدود البرنامج.و أصبح بالإمكان دائماً تخزين محتويات الملف على القرص باستخدام أمر مناسب ضمن البرنامج.

إذا كان الوسيط الثاني للإجراء ASSIGN هو سلسلة فارغة فإن هذا الإجراء سيربط متحول الملف مع وحدة الإدخال و الإخراج الأساسية consol لذلك سيربط الملف fconsol مع الشاشة عبر الاستدعاء التالي:

Assign(FConsol, ' ');

بعد عملية ربط الملف بالبيئة الخارجية نحتاج الآن إلى جعل الملف جاهزاً للاستخدام من قبل البرنامج و ذلك باستخدام الإجراءين RESET و REWRITE. (في لغة باسكال القياسية و بعض لغات باسكال قد دمج الإجراء ASSIGN مع الإجراءين RESET و REWRITE).

إنشاء ملف Creating a New File:

ينشئ الإجراء REWRITE ملفاً جديداً حيث يبنى الملف الجديد بواسطة الكتابة عليه . وبعد ذلك نستطيع قراءة المعلومات من هذا الملف متى نريد. ولكن الجدي التي جاءت به لغة تربو باسكال هو إمكانية إضافة المعلومات إلى ملف موجود أصلاً فيه بعض المعلومات.

مثال an example: يرينا المثال التالي كيفية إنشاء ملف نصي جديد و كيفية قيمة واحدة مرتين على هذا الملف.

code

```
program test;
const filename='c:\zwzw.wzw';
var samplefile:text;
begin
assign(samplefile,filename);
rewrite(samplefile);
writeln(samplefile,397.5);
writeln(samplefile,397.5:10:5);
writeln('Done');
readln
end.
```

يستخدم الإجراءان مسبقا التعريف write , writeln من أجل كتابة ما نريد من معلومات ضمن الملف و ذلك بعد الإشارة إلى أن كتابة هذه المعلومات سوف تتم على ملف و ذلك عبر تمرير متحول الملف كمتحول وسيطي أول لهذين الإجراءين و من ثم تمرير المعلومات بعد ذلك ضمن متحولات وسيطية تالية فمثلاً استدعاء الإجراء WRITELN قبل الأخير ضمن البرنامج السابق سوف يكتب القيمة (397.5) على الملف samplefile .

إذا سردنا محتويات لدليل directory من الملفات بعد تنفيذ هذا البرنامج سوف نرى الملف zwzw.wzw ضمن السواقة c هما وبحجم 0 بايت .

إذاً ماذا حدث للقيمة (397.5) التي كتبت مرتين في هذا الملف ؟؟؟ في الحقيقة هذه القيمة ضاعت بسبب عدم تخزين محتويات الملف قبل إنهاء البرنامج وعلينا استخدام الإجراء close قارن الملف zwzw.wzw بعد تنفيذ البرنامج التالي مع الملف الناتج عن البرنامج السابق:

code

```
program test;
```

```
const filename='c:\zwzw.wzw';
var samplefile:text;
begin
assign(samplefile,filename);
rewrite(samplefile);
writeln(samplefile,397.5);
writeln(samplefile,397.5:10:5);
close(samplefile);
writeln('Done');
readln
end.
```

إذا نظرنا إلى محتويات الملف zwzw.wzw بواسطة محرر نصوص فسوف نرى العدد (397.5) ضمن الملف. لأن استدعاءنا للإجراء close قد فرغ الذاكرة الوسيطية (buffer) المخصصة للملف و خزنها في القرص قبل إغلاق الملف. و محتويات هذا الملف ستكون كالتالي:

```
3.9750000000E+02
 397.50000
```

أن التصريح عن المتحولات ضمن البرنامج جعل المترجم compiler يخصص مساحة من الذاكرة لمتحول من النوع text وهذا المتحول يخزن داخلياً كسجل يحتوي على معلومات متعددة عن الملف وهذا السجل يحتوي أيضاً على ذاكرة وسيطية buffer تكتب المعلومات فيها أو تنقل المعلومات منها إلى الملف . ويجري تفريغ هذه الذاكرة الوسيطية في الملف الموجود على القرص قبل إغلاق هذا الملف.

بعد أن صرحنا عن المتحول samplefile على أنه متحول من النوع ملف نصي text علينا القيام بعدو أمور:

١. كتابة عبارة assign لتربط متحول الملف samplefile باسم الملف filename فمتحول الملف يُمرر على انه متحول وسيطي أول واسم الملف يمرر كمتحول وسيطي ثاني.
٢. إنشاء وفتح ملف جديد بواسطة الإجراء rewrite و الذي يأخذ متحولاً وسيطياً واحداً هو متحول الملف. فاستدعاءنا للإجراء rewrite هذا سوف يفتح الملف الذي ربطه اسمه بمتحول الملف samplefile الموجود ضمن البرنامج.ويجب الانتباه هنا إلى أن الإجراء rewrite ينشئ دائماً ملفاً جديداً فإذا كان هناك ملف قديم يحمل نفس اسم الملف الجديد فإن المعلومات ضمن الملف القديم سوف تضيع ويستبدل الملف القديم بالملف الجيد الفارغ.
٣. كتابة المعلومات في هذا الملف باستخدام روتينات الإدخال /الإخراج (I/O) المسبقة التعريف أعني write و writeln و هذان الإجراءان لهما نفس الصيغة الكتابية
٤. بعد ما كتبنا ما نريد من معلومات على الملف علينا الآن إغلاق الملف بواسطة استدعاء الإجراء close حيث يتم تفريغ الذاكرة الوسيطية المرتبطة بالملف بواسطة كتابة محتوياتها على القرص أي تخزين هذه المعلومات ومن ثم يغلق هذا الملف.

الذاكرة الوسيطية لملف (file buffer) هي بالتعريف عبارة عن نسق محدود الحجم يعادل 128 bytes للملفات النصية في لغة تربو باسكال .تكتب المعلومات المراد تخزينها في الملف في هذه الذاكرة الوسيطية وفي حال امتلاء هذه الذاكرة قبل إغلاق الملف تفرغ هذه الذاكرة في الملف و تعود من جديد لتكون مستعدة لتلقي المعلومات.

مثال an example:

يكتب البرنامج التالي 125 قيمة عشوائية ضمن ملف:

```
code
program test;
const
maxtrials=125;
filename='c:\zwzw.wzw';
var samplefile:text;
index:integer;
begin
randomize;
assign(samplefile,filename);
```

```
rewrite(samplefile);
for index:=1 to maxtrials do
begin
write(samplefile,random:10:5);
if index mod 5 =0 then
writeln(samplefile);
end;
close(samplefile);
writeln('Done');
readln
end.
```

الإجراء ;writeln(samplefile) مكافئ تماما للإجراء ;writeln الذي يسبب انتقالاً إلى سطر جديد على الشاشة.أما الاستدعاء ;writeln(samplefile) فيتقل إلى سطر جديد في الملف.

محتويات الملف zwzw.wzw بعد تنفيذ عشوائي للبرنامج:

0.22389	0.10634	0.03516	0.42838	0.22141
0.97770	0.24730	0.94828	0.02989	0.63034
0.38868	0.18200	0.51128	0.99480	0.96634
0.68601	0.30416	0.29503	0.00504	0.85922
0.52878	0.51094	0.04765	0.38258	0.17274
0.24433	0.17222	0.74178	0.89170	0.65996
0.38496	0.62972	0.20950	0.42375	0.99859
0.27582	0.35214	0.80988	0.85546	0.99895
0.19980	0.39872	0.50469	0.53081	0.09149
0.03751	0.69286	0.90682	0.46663	0.98548
0.77751	0.97108	0.57782	0.41558	0.92291
0.79266	0.75935	0.69439	0.18733	0.53975
0.98577	0.11234	0.86146	0.60879	0.70480
0.98587	0.66698	0.05527	0.17906	0.30367
0.67503	0.62327	0.82100	0.33866	0.05365
0.92037	0.25531	0.89769	0.00380	0.53512
0.09331	0.13078	0.49316	0.14356	0.65354
0.07186	0.64721	0.90207	0.11312	0.33279
0.90683	0.67309	0.82405	0.62260	0.31198
0.88079	0.77900	0.01375	0.52068	0.04347
0.46084	0.12741	0.42268	0.27962	0.56038
0.65950	0.45731	0.33444	0.29657	0.91946
0.99474	0.92381	0.32373	0.30355	0.39015
0.44244	0.45995	0.87468	0.57004	0.33324
0.72123	0.47044	0.26026	0.66461	0.39565

ملاحظة:التابع random بدون تمرير متحولات له يولد قسم عشوائية ضمن المجال [1..0.0]

فحص نتائج العمليات على ملفاتنا checking the outcome of your file operation:

قد يفشل البرنامج السابق لعدة أسباب منها عدم إمكانية إنشاء الملف . افترض مثلا عدم وجود مساحة فارغة في القرص لا تكفي لإنشاء الملف عندئذ الملف الجديد لن يظهر إلى الوجود .

لغة تربو باسكال تفحص كل استدعاءات الإجرائيات التي تتعامل مع الأقراص مثل: rewrite فإذا كان الإجراء غير قادر على إنشاء الملف فإن هذا الخطأ سوف يُرصد و سوف ينهي تنفيذ البرنامج وتظهر رسالة خطأ.

إن عملية الفحص التلقائي وخاصية الإنهاء automatic checking and termination تحمينا من بعض أنواع الأخطاء التي قد تسبب مشاكل في مراحل متقدمة من البرنامج و لكن عملية الإنهاء التلقائية لها سيئة أيضاً وهي أن تجعل البرنامج أقل متانة و ذلك بسبب حدوث أي خطأ ضمن الإجراءات التي تتعامل مع الملفات.

تملك لغة تربو باسكال توجيهات للمترجم لاختيار حالة الفحص التلقائي تلك حيث نستطيع أن نفحص نحن نتائج استدعاء الإجراءات تاركين عملية الإنهاء التلقائي في حال حدوث خطأ ما وذلك باستخدام توجيه المترجم التالي {$I -} و الذي لن ينهي البرنامج في حال عدم نجاح أي عملية من عمليات الدخل أو الخرج وسيتكرر نفس الخطأ في كل تنفيذ للبرنامج .
يمكننا إجراء عملية الفحص تلك من خلال التابع مسبق التعريف IOResult و الذي يعيد قيمة من النوع word تمثل نتيجة آخر عملية إدخال أو إخراج . فإذا تمت هذه العملية بنجاح فإن هذا التابع يعيد القيمة 0 و إلا فإنه سوف يعيد شفرة الخطأ الحاصل.
حتى يعيد التابع IOResult قيمة لها معنى يجب أن تكون حالة الفحص هي الفحص غير التلقائي (الفحص اليدوي) فإذا حصل أي خطأ أثناء التنفيذ عندئذ سيعيد هذا التابع شفرة الخطأ. ويجب الانتباه إلى أن التابع IOResult يعيد شفرة آخر عملية إدخال أو إخراج لذلك علينا الانتباه إلى موضع استدعاء هذا الإجراء وتكرار عملية استدعاء هذا الإجراء إذا تطلب الأمر ذلك بعد كل عملية إدخال أو أخراج.
حتى نعود إلى حالة الفحص التلقائي نكتب توجيه المترجم المعاكس أي {$I +} .
يوضع البرنامج التالي لنا كيفية استخدام توجيه المترجم السابق مع ملاحظة أن البرنامج التالي لن يتوقف تنفيذه أبداً مع أي خطأ أنشئ الملف أم لم ينشئ الملف :

```
                         code
program test;
const
filename='c:\zwzw.wzw';
var samplefile:text;
procedure wait;
begin
writeln(' press Enter to continue... ');
readln;
end;
begin
assign(samplefile,filename);
(*$I-*)
rewrite(samplefile);
(*$I+*)
if(IOResult=0)then
writeln('successful')
else
writeln('could not create file ',filename,'!');
wait;
writeln('Done');
readln
end.
```

يشرح البرنامج السابق طريقة تغيير حالة الفحص المدخل/المخرج حيث جرى تعديل حالة الفحص قبل استدعاء الإجراء rewrite وقبل بعض الإجراءات التي تتعامل مع الملفات . و بعد هذه الاستدعاءات أعيدت حالة الفحص إلى ما كانت عليه باستخدام التوجيه {$I+} .

```
                         code
program test;
function fmade(var thefile:text;fname:string):boolean;

begin
end;
const
maxtrials=125;
filename='c:\zwzw.wzw';
var samplefile:text;
procedure wait;
```

```
begin
writeln(' press Enter to continue... ');
readln;
end;
begin
assign(samplefile,filename);
(*$I-*)
rewrite(samplefile);
reset(samplefile);
writeln(samplefile,'khaled');
if(IOResult=0)then
writeln('successful')
else
writeln('could not create file ',filename,'!');

wait;
close(samplefile);
writeln('Done');

readln
end.
```

لن يسبب البرنامج السابق حدوث أي خطأ ترجمة أو خطأ أثناء التفيذ.

إجراء لإنشاء الملف بأمان A Safe file creation procedure:

إن تتابع العبارات وتوجيهات المترجم في المثال السابق هو تتابع شهير يستخدم عادة عند التعامل مع الملفات لذلك سوف ننشئ تابعاً نستخدمه دائماً في معالجة الملفات:

```
Code
function fmade(var thefile:text;fname:string):boolean;
begin
assign(thefile,filename);
(*$I-*)
rewrite(thefile);
(*$I+*)
if IOResult=0 then
fmade:=true
else
fmade:=false
end;
```

سوف نستدعي هذا التابع عندما نريد إنشاء ملف ونكون قد قصرنا من شفرة البرنامج حيث استعضنا عن عدة أسطر بسطر واحد و جعلنا بذلك برامجنا أكثر مرونة وقوة.في حال حدوث خطأ ما فإن البرنامج يمكن أن يقوم بعمل مناسب عوضاً عن حدوث فشل في تنفيذه.

نلاحظ من خلال التابع السابق كيفية تمرير الملفات النصية كوسطاء إلى الروتينات و عملية التمرير تتم وفق نفس الصيغة الكتابية لتمرير الوسطاء الأخرى و هي:

< نوع معطيات محدد>":"<مميز>

ينبغي للوسطاء التي تنتمي إلى نوع المعطيات ملف أن تمرر مرجعياً أي أن الروتين لن ينسخ محتويات الملف لأن الروتين لا يعرف كم سيغدو حجم الملف لذلك لا يستطيع حجز حجرات محددة لمتحول الملف بل يستخدم نفس الحجرات التي خصصها المترجم لهذا الملف ولذلك لا تمرر الملفات إلا مرجعياً .

فتح ملف موجود opening an existing file:

يمكننا استدعاء الإجراء reset عوضاً عن rewrite إذا كان الملف المطلوب موجوداً أو أردنا الحفاظ على المعلومات المخزنة فيه .فالإجراء reset يفتح الملف المحدد ضمن متحوله الوسيطي الأول مفترضاً أن هذا الملف موجود وسوف يفتح للقراءة .

لدينا البرنامج التالي يقرأ محتويات الملف zwzw.wzw الذي يحوي القيم العشوائية ويحلل هذه القيم:

Code

```
program test;
const maxtrials=25;
maxperline=5;
filename='c:\zwzw.wzw';
type stats=array[0..4]of real;
var samplefile:text;
index,whichcell:integer;
vals:stats;
result:real;
function fopened(var thefile:text;fname:string):boolean;
begin
assign(thefile,fname);
(*$I-*)
reset(thefile);
(*$I+*)
if (IOResult=0)then
fopened:=true
else
fopened:=false
end;
procedure init(var v:stats);
var i:integer;
begin
for i:=1 to 4 do
v[i]:=0.0;
end;
procedure show(v:stats;va:integer);
var i:integer;
begin
for i:=1 to 4 do
writeln(i,': ',v[i]/va:10:5);
writeln(5,': ',v[0]/va:10:5);
end;
begin
randomize;
init(vals);
if fopened(samplefile,filename)=true then
begin
for index:=1 to maxtrials do
begin
read(samplefile,result);
whichcell:=index mod maxperline;
vals[whichcell]:=vals[whichcell]+result;
write(vals[whichcell]:10:5);
if index mod maxperline=0 then
```

```
writeln;
end;
show(vals,maxtrials div maxperline);
close(samplefile);
end
else
writeln('could not open file ',filename);
readln;
end.
readln;
end.
```

محتويات الملف zwzw.wzw هي القيم العشوائية:

```
0.22389 0.10634 0.03516 0.42838 0.22141
0.97770 0.24730 0.94828 0.02989 0.63034
0.38868 0.18200 0.51128 0.99480 0.96634
0.68601 0.30416 0.29503 0.00504 0.85922
0.52878 0.51094 0.04765 0.38258 0.17274
0.24433 0.17222 0.74178 0.89170 0.65996
0.38496 0.62972 0.20950 0.42375 0.99859
0.27582 0.35214 0.80988 0.85546 0.99895
0.19980 0.39872 0.50469 0.53081 0.09149
0.03751 0.69286 0.90682 0.46663 0.98548
0.77751 0.97108 0.57782 0.41558 0.92291
0.79266 0.75935 0.69439 0.18733 0.53975
0.98577 0.11234 0.86146 0.60879 0.70480
0.98587 0.66698 0.05527 0.17906 0.30367
0.67503 0.62327 0.82100 0.33866 0.05365
0.92037 0.25531 0.89769 0.00380 0.53512
0.09331 0.13078 0.49316 0.14356 0.65354
0.07186 0.64721 0.90207 0.11312 0.33279
0.90683 0.67309 0.82405 0.62260 0.31198
0.88079 0.77900 0.01375 0.52068 0.04347
0.46084 0.12741 0.42268 0.27962 0.56038
0.65950 0.45731 0.33444 0.29657 0.91946
0.99474 0.92381 0.32373 0.30355 0.39015
0.44244 0.45995 0.87468 0.57004 0.33324
0.72123 0.47044 0.26026 0.66461 0.39565
```

خرج البرنامج هو:

```
0.22389 0.10634 0.03516 0.42838 0.22141
1.20159 0.35364 0.98344 0.45827 0.85175
1.59027 0.53564 1.49472 1.45307 1.81809
2.27628 0.83980 1.78975 1.45811 2.67731
2.80506 1.35074 1.83740 1.84069 2.85005
3.04939 1.52296 2.57918 2.73239 3.51001
3.43435 2.15268 2.78868 3.15614 4.50860
3.71017 2.50482 3.59856 4.01160 5.50755
```

```
  3.90997  2.90354  4.10325  4.54241  5.59904
  3.94748  3.59640  5.01007  5.00904  6.58452
  4.72499  4.56748  5.58789  5.42462  7.50743
  5.51765  5.32683  6.28228  5.61195  8.04718
  6.50342  5.43917  7.14374  6.22074  8.75198
  7.48929  6.10615  7.19901  6.39980  9.05565
  8.16432  6.72942  8.02001  6.73846  9.10930
  9.08469  6.98473  8.91770  6.74226  9.64442
  9.17800  7.11551  9.41086  6.88582 10.29796
  9.24986  7.76272 10.31293  6.99894 10.63075
 10.15669  8.43581 11.13698  7.62154 10.94273
 11.03748  9.21481 11.15073  8.14222 10.98620
 11.49832  9.34222 11.57341  8.42184 11.54658
 12.15782  9.79953 11.90785  8.71841 12.46604
 13.15256 10.72334 12.23158  9.02196 12.85619
 13.59500 11.18329 13.10626  9.59200 13.18943
 14.31623 11.65373 13.36652 10.25661 13.58508
1:   0.57265
2:   0.46615
3:   0.53466
4:   0.41026
5:   0.54340
```

هذا البرنامج يقرأ القيم العشوائية المخزنة في الملف zwzw.wzw بطريقة تتابعية أي أن القيمة الثانية سوق تقرأ بعد القيمة الأولى والقيمة الثالثة تقرأ بعد القيمة الثانية وهكذا دواليك.... و هذه الطريقة التتابعية هي الطريقة الوحيدة التي يمكن من خلالها الوصول إلى المعلومات المخزنة ضمن الملفات النصية text files لذلك تدعى الملفات النصية ببنى الوصول التتابعي sequential –access- structures في حين أن الأنساق أو حتى بعض أنواع الملفات تدعى ببنى الوصول العشوائي random access structures .

إذا البرنامج السابق يقرأ القيم المخزنة ضمن الملف و يجمع قيم كل عمود من أعمدة الملف الخمسة و يخزن ناتج عملية الجمع هذه ضمن نسق مؤلف من خمس حجرات من نوع المعطيات عدد حقيقي. و أثناء قراءة وجمع هذه القيم يظهر البرنامج القيم الحالية لحجرات النسق و التي تحتوي دائما على ناتج جمع هذه القيم . ومن ثم يحسب البرنامج معدل هذه القيم الواقعة طبعا ضمن عمود واحد.

لدينا التابع fopened وهو إجراء للتأكد من وجود الملف المطلوب و فتحه في حال وجود الملف وبما أن الملف المطلوب موجود أصلاً لذلك استخدمنا الإجراء reset الذي يستخدم لفتح ملفات موجودة للقراءة و تدعى الملفات المفتوحة بهذا الإجراء بملفات الخرج output files أما الإجراء init يستخدم لتهيئة النسق و الإجراء show يستخدم لإظهار النتائج على الشاشة.

يرينا البرنامج كيفية استدعاء الإجراء read ضمن البرنامج الرئيسي كيفية الحصول على المعلومات من الملفات النصية حيث أن الإجراءين read و readlnم سبقي التعريف يتعاملان مع الملفات بنفس الطريقة التي تتعامل بهار وتينات الخرج وأعني الإجراءين write و writeln فالمتحول الوسيطي الأول يجب أن يكون متحول ملف.

فإذا طلبنا مثلاً من الإجراء read قراءة قيمة من النوع integer فإن هذا الإجراء سوف يفحص القيم الموجودة في الملف حتى يصل إلى أول قيمة صحيحة يصادفها ويتوقف الإجراء عند ذلك الوقع . فإذا استدعي مرة أخرى فسيتابع قراءته من هذه النقطة التي توقف عندها.أما إذا صادف هذا الإجراء رمز نهاية السطر end of line فإن الإجراء ببساطة فسوف ينتقل إلى سطر جديد. علما أنه يمكننا فحص هذا الرمز بواسطة الإجراء EOLN و ذلك باستخدام الاستدعاء التالي:

EOLN(samplefile);

سوف يعيد الاستدعاء السابق القيمة true إذا صادف رمز نهاية السطر. ويوجد أيضاً تابع مسبق التعريف EOF يأخذ متحولا وسيطياً واحدا هو متحول ملف يعيد قيمة بوليانية هي true إذا وصل إلى نهاية الملف end-of-file و إلا يعيد القيمة FALSE .

الإجراء read مقابل الإجراء readln (read versus readln):

البرنامج السابق مطابق للبرنامج التالي إلا أن استبدلنا الإجراء read الإجراء readln ضمن حلقة for في البرنامج الرئيسي:

Code

```
program test;
const maxtrials=25;
maxperline=5;
filename='c:\zwzw.wzw';
type stats=array[0..4]of real;
var samplefile:text;
index,whichcell:integer;
vals:stats;
result:real;
function fopened(var thefile:text;fname:string):boolean;
begin
assign(thefile,fname);
(*$I-*)
reset(thefile);
(*$I+*)
if (IOResult=0)then
fopened:=true
else
fopened:=false
end;
procedure init(var v:stats);
var i:integer;
begin
for i:=1 to 4 do
v[i]:=0.0;
end;
procedure show(v:stats;va:integer);
var i:integer;
begin
for i:=1 to 4 do
writeln(i,': ',v[i]/va:10:5);
writeln(5,': ',v[0]/va:10:5);
end;
begin
randomize;
init(vals);
if fopened(samplefile,filename)=true then
begin
for index:=1 to maxtrials do
begin
readln(samplefile,result);
whichcell:=index mod maxperline;
vals[whichcell]:=vals[whichcell]+result;
write(vals[whichcell]:10:5);
if index mod maxperline=0 then
writeln;
end;
show(vals,maxtrials div maxperline);
close(samplefile);
end
```

```
else
writeln('could not open file ',filename);
readln;
end.
readln;
end.
```

خرج البرنامج هو:

```
  0.22389  0.97770  0.38868  0.68601  0.52878
  0.46822  1.36266  0.66450  0.88581  0.56629
  1.24573  2.15532  1.65027  1.87168  1.24132
  2.16610  2.24863  1.72213  2.77851  2.12211
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
  2.62694  2.90813  2.71687  3.22095  2.84334
1:  0.10508
2:  0.11633
3:  0.10867
4:  0.12884
5:  0.11373
```

تُرى ما هي التغييرات التي حصلت في خرج البرنامج بعد هذا التعديل ؟؟؟؟ إن استدعاء الإجراء readln كما تعلم ينتقل إلى سطر جديد بعد قراءة كل قيمة لذلك سوف تقرأ قيمة واحدة من ل خمس قيم موجودة في السطر وبالتالي بعد خمس وعشرين استدعاء للإجراء readln سيصل البرنامج إلى نهاية الملف و الذي يحتوي أصلا على خمسة وعشرين سطراً. وعندها سيعيد الإجراء readln القيمة 0 عند استدعائه و لن تتغير قيم حجرات النسق.
تكرر السطر الخامس في هذا الخرج عشرين مرة و ذلك بسبب وصول الإجراء readln إلى نهاية الملف و بالتالي لم تعد تتغير قيم حجرات النسق لذلك نقول :هنالك فرق كبير في استخدام الإجراءين read و readln للحصول على معلومات من الملفات. حيث يستخدم الإجراء read إذا أردنا قراءة عدة قيم ضمن سطر واحد لأن استخدام الإجراء readln في هذه الحالة سوف يسبب فقدان معظم هذه القيم لأنه يقرأ أول قيمة و ينتقل إلى سطر جديد.

مثال:استبدال رموز الجدولة TAB من ملف example :removing tabs from a file:

ينسخ البرنامج التالي أسطر الملف إلى ملف آخر مع استبدال كل رمز جدولة tab (والذي شفرة ASCII المقابلة له هي 9) بعدد مناسب من الفراغات. ويبرز البرنامج التالي سلوك الإجراء read عند قراءته للرموز و write :

Code

```
program test;
const smallcycle=50;
largecycle=2500;
var source,log:text;
nrchread:longint;
sname,lname:string;

function fmade(var thefile:text;fname:string):boolean;
begin
assign(thefile,fname);
(*$I-*)
rewrite(thefile);
(*$I+*)
if (IOResult=0)then
fmade:=true
else
fmade:=false
end;

function fopened(var thefile:text;fname:string):boolean;
begin
assign(thefile,fname);
(*$I-*)
reset(thefile);
(*$I+*)
if (IOResult=0)then
fopened:=true
else
fopened:=false
end;
procedure getstring(message:string; var value:string);
begin
write(message,' ');
readln(value);
end;
procedure showprogress(val,small,large:longint);
const markerch='.';
begin
if(val mod small=(small-1))then
write(markerch);
if(val mod large=(large-1))then
writeln;
end;
procedure detabfile(var infile,outfile:text; var count:longint);
const tab=#9;
cr=#13;
lf=#10;
ctrlz=#26;
```

```
tabskip=8;
var ch:char;
index:integer;
begin
index:=1;count:=0;
while not eof(infile) do
begin
read(infile,ch);
case ch of
tab:
{repeat   }
write(outfile,' ');
{inc(index);
until(index mod tabskip=1);}
cr,lf:begin
write(outfile,ch);
index:=1;
end;
ctrlz: ;
else
begin
write(outfile,ch);
inc(index);
end;
end;
inc(count);
showprogress(count,smallcycle,largecycle);
end;{wghile not eof }
end;
procedure wait;
begin
writeln('press enter to contiune. ');
readln;
end;
begin {main}
getstring('source fil name?',sname);
getstring('log file name?',lname);
if(fopened(source,sname)=true)and(fmade(log,lname)=true)then
begin
detabfile(source,log,nrchread);
writeln;
writeln(nrchread,'chars read');
close(source);
close(log);
end;
wait;
end.
```

يطلب البرنامج من المبرمج أو المستخدم إدخال اسمين لملفين الأول منهما يشير إلى الملف الموجود و الذي يحتوي على رموز الجدولة tab وهذا الملف infile سوف يقرأ رمزاً رمزاً و ستجري التعديلات اللازمة ضمنه و من ثم تنسخ إلى الملف الثاني أي outfile.

الإجراء detabfile هو لب لبرنامج السابق حيث يقرأ هذا الروتين رمزاً من الملف infile فإذا كان هذا الرمز هو رمز الجدولة TAB فسوف يستبدله بفراغ واحد فقط أما إذا كان الرمز هو رمز إرجاع CR (carriage return) أو رمز التغذية السطرية (linefeed) فسوف ينسخ الرمز كما هو إلى الملف الثاني أما الرموز الأخرى في الملف سوف تنسخ كما هي إلى الملف الثاني ما عدا الرمز ctrl-z والذي شفرة ASCII المقابلة له 26 فسوف يعني نهاية الملف وبالتالي إنهاء عملية النسخ.

يخبر الإجراء showprogress المستخدم user بأن البرنامج في حالة عمل و هذا الإجراء ضروري جداً في حال البرامج التي تقوم بأعمال تتطلب فترات زمنية طويلة نسبياً لأن البرنامج في هذه الفترات لا يظهر أي خرج على الشاشة لذلك يظهر الإجراء بعض النقاط dots على الشاشة. و في الحقيقة يقوم الإجراء بكتابة نقطة على الشاشة بعد قراءة البرنامج لخمسين رمزاً من الملف infile و ينقل هذا الإجراء إلى سطر جديد بعد أن يقرأ البرنامج 2500 رمزاً من الملف نفسه.

يعيد الإجراء detabfile الرموز التي يقرأها من الملف القديم وتمرر هذه القيم كوسيط أول إلى الإجراء showprogress .

و مما يلاحظ هنا أن الملفات قد مررت إلى الإجراء detabfile بنفس الطريقة التي تمرر فيها أية وسطاء إلى الروتينات ويجب أن تُسبق وسطاء الملفات بالمميز var أو بمعنى آخر لا تمرر الملفات إلا مرجعياً passed by reference

قراءة رموز مميزة من الملف reading single characters from a file:

يقرأ البرنامج السابق المعلومات من الملف رمزا رمزا باستخدام الإجراء مسبق التعريف READ و هذا الإجراء يقرأ رمزا واحدا من الملف المحدد له ضمن متحوله الوسيطي الأول و يعيد الرمز المقروء ضمن متحوله الوسيطي الثاني والإجراء read يقرأ معظم الرموز بهذه الطريقة إلا أنه توجد بعض الرموز التي يجب معاملتها معاملة خاصة كرمز الإرجاع مثلاً. فعندما يُقرأ هذا الرمز فإن نتيجة القراءة تلك تعتمد على قيمة التابعين EOLN و EOF هل هي TRUE أم FALSE .

هذا يعني أن النتيجة تلك نعتمد على موقع الرمز هل هو في نهاية السطر أم في نهاية الملف.

فإذا وصلنا إلى نهاية الملف أي أن استدعاء التابع EOF(infile) يعيد القيمة true فسيعيد الإجراء read القيمة ctrl-Z ضمن وسيطه الثاني أما إذا أعاد استدعاء التابع التالي EOLN(infile) القيمة true فسيعيد الإجراء read القيمة cr أي رمز الإرجاع carriage return و الذي شفرة ascii النقابلة له هي 13 وينتقل البرنامج إلى بداية سطر جديد ضمن الملف.

إضافة المعلومات إلى ملف موجود adding to an existing file:

لقد ذكرنا سابقاً أن الإجراء reset يفتح ملفاً موجوداً للقراءة و الإجراء rewrite ينشئ ملفاً جديداً للكتابة . وذكرنا أيضاً أننا لا نستطيع الكتابة على ملف فتح بواسطة الإجراء reset ولا نستطيع أيضاً قراءة المعلومات من ملف فتح بواسطة الإجراء rewrite .لكننا قد نحتاج في بعض الأحيان إلى مفتح ملف موجود لإضافة بعض المعلومات إليه لذلك نستخدم الإجراء append الذي يأخذ متحولاً وسيطياً واحداً من النوع text حيث يفتح الملف و يضع مؤشر الملف عند نهاية هذا الملف استعداداً لتلقي المعلومات التي ستضاف إليه . فإذا لم يجد الإجراء append الملف المطلوب فسيعطيننا خطأ في تنفيذ البرنامج .و في حال استدعاء الإجراء append لملف مفتوح فإن هذا الإجراء سوف يُغلق هذا الملف و من ثم يفتحه مرة أخرى. و هذه العملية تعد خطرة و قد تؤدي إلى ضياع بعض المعلومات لذلك لا ننصح أبداً باستدعاء الإجراء append لملف مفتوح.

يضيف البرنامج التالي 125 قيمة عشوائية أخرى إلى الملف zwzw.wzw باستخدام استدعاء الإجراء append:

code

```
program test;
const maxtrials=125;
maxperline=5;
filename='c:\zwzw.wzw';
var samplefile:text;
index:integer;

function fappended(var thefile:text;fname:string):boolean;
begin
assign(thefile,fname);
(*$I-*)
append(thefile);
```

```
(*$I+*)
if (IOResult=0)then
fappended:=true
else
fappended:=false
end;

procedure wait;
begin
writeln('press enter to contiune. ');
readln;
end;
begin
randomize;
if fappended(samplefile,filename)=true then
begin
for index:=1 to maxtrials do
begin
write(samplefile,random:10:5);
if index mod maxperline =0 then
writeln(samplefile);
end;
close(samplefile);
end
else
writeln('file ', filename,' could not be opened. ');
wait;
end.
```

العمليات المطبقة على الملفات النصية operations on text files:

يمكننا تلخيص العمليات الأساسية التي يمكننا القيام بها على الملفات النصية كما يلي:

١. يجب استخدام الإجراء assign لربط متحول من النوع text مع ملف على القرص .و ذلك بواسطة تمرير متحول الملف النصي كمتحول وسيطي أول لهذا الإجراء و تمرير اسم الملف الموجود على القرص كمتحول وسيطي ثاني.
٢. يستخدم الإجراء rewrite لإنشاء ملف جديد و فتحه للكتابة فقط. وفي حال وجد ملف يحمل نفس الاسم فإن الملف القديم سوف يُستبدل بالملف الجديد و تضيع المعلومات الموجودة فيه.
٣. يمكم استخدام الإجراء reset لإنشاء ملف لفتح ملف موجود للقراءة . يجب أن يكون الملف موجودا وإلا سوف يفشل تنفيذ البرنامج و يعطينا رسالة خطأ في تنفيذ البرنامج طبعا ما لم نغير حالة فحص الدخل و الخرج إلى حالة الفحص اليدوي بدلا من الفحص التلقائي.
٤. نستطيع استخدام append لفتح ملف موجود و كتابة بعض المعلومات في نهايته.
٥. يستخدم الإجراء close لإغلاق الملف الذي قد فتحناه بأحد الإجراءات السابقة حيث يُفرغ هذا الإجراء الذاكرة الوسيطية في الملف الموجود على القرص و من ثم يغلق الملف أي فعلياً يخزن الملف و يحفظه من الضياع.
٦. يستخدم الإجراءان read و readln لقراءة المعلومات من الملف النصي و هذه المعلومات يمكن أن تكون من أي نوع معطيات بسيط عدا نوع المعطيات التعدادي و نوع المعطيات بولياني و يمكنهما أيضاً قراءة سلاسل رمزية من الملف.
٧. يستخدم الإجراءان write و writeln لكتابة المعلومات في الملفات النصية و هذه المعلومات يمكن أن تكون من أي نوع معطيات بسيط عدا نوع المعطيات التعدادي و يمكنهما أيضاً كتابة سلاسل رمزية على الملف.
٨. يمكننا استخدام التابع IOResult لفحص نتيجة عملية الدخل أو الخرج هي الفحص اليدوي أي : {$I-} و {$I} اعتمادا على القيمة هذا التابع يمكن للبرنامج اتخاذ خطوات مناسبة لضمان سلامة تنفيذ المهمة الموكلة للبرنامج فالقيمة 0 إذا أعادها هذا التابع تعني أن عملية الإدخال أو لإخراج قد تمت بنجاح.
٩. نستطيع استخدام التابع EOF لتحديد وصول البرنامج إلى نهاية الملف المحدد أم لا.
١٠. نستطيع استخدام التابع EOLN لتحديد وصول البرنامج إلى نهاية السطر الحالي أم لا.

إعادة تسمية و حذف الملفات renaming and erasing files:

تتطلب عملية إدارة الملفات النصية التمكن من إعادة تسمية ملف ما أو حذف هذا الملف إذا تتطلب الأمر و بدون العودة إلى محث النظام DOS لذلك زودتنا لغة تربو باسكال بالإجراءين rename و erase .
فالإجراء rename مهمته إعادة تسمية ملف ما وفيما يلي مثال على الإجراء السابق:

```
Assign(filevariable,'file.old');
Rename(filevariable.'file.new');
```

إما الإجراء erase فيحذف الملف من القرص و ذلك أيضاً بعد ربط هذا الملف بمتحول لهذا الملف وذلك وفق الطريق التالية:

```
Assign(filevariable,'file.old');
Erase(filevariable);
```

التابعان seekEof و seekEoln seekEof and seekeoln functions :

تزودنا لغة تربو باسكال بالتابعين **seekEof و seekEoln** اللذين يقومان بما تقوم به نظيراهما التابعان EOF و EOLN .حيث يعيد التابع seekEof القيمة TRUE إذا وصل مؤشر الملف إلى نهاية الملف أم التابع seekEOLN فيعيد القيمة TRUE إذا وصل مؤشر الملف إلى نهاية سطر ما . لكن الفرق بين هذين التابعين و التابعين السابقين هو أن التابعين seekEof و seekEoln يتجاهلان فحص رموز التحكم في شفرة ASCII و التي تتراوح شفرتها بين 0 و32 لذلك فإن التابع seekeof يعيد القيمة true حتى ولو بقي بعض الرموز في الملف ما دامت هذه الرموز هي رموز تحكم أو رمز الفراغ.

الذاكرة الوسيطية للملف file buffer:

إن عملية الكتابة أو القراءة من القرص تعتبر بطيئة نسبة إلى سرعة إنجاز العمليات في الحاسب لذلك تخصص لغة تربو باسكال مساحة من الذاكرة تسميها الذاكرة الوسيطية buffer تستخدمها كمخزن مؤقت للمعلومات التي ستخزن على القرص أو التي ستقرأ منه.
وقد ذكرنا سابقاً أن حجم هذه الذاكرة هو 128 بايت فإذا طلبنا قراءة 10 بايتات مثلاً من ملف فسيُقرأ منه 128 بايتا و يوضع في الذاكرة الوسيطية تحسباً لطلب قراءة آخر مما يُسرع عملية الوصول للمعلومات .
ولكن قد يكون هذا الحجم صغيراً عندما نتعامل مع ملفات كبيرة تحتاج إلى تبادل معلومات كثيرة مع القرص لذلك زودتنا لغة تربو باسكال بإمكانية تحديد حجم الذاكرة الوسيطية التي نريد و ذلك عبر استدعاء الإجراء settextbuf الذي يحدد حجم الذاكرة الوسيطية المخصصة للملف النصي.
لتأخذ جزء البرنامج التالي الذي يربط ملفاً نصياً بملف على القرص ويحدد له حجم الذاكرة الوسيطية على إنه 1 كيلو بايت كحد أعظمي كما يلي:

```
Var textfile:text;
 Buffer:array[1..1024] of byte;
Begin
Assign(textfile,'c:\toto.soso');
Settextbuf(textfile,buffer);
Reset(textfile);
```

يُخصص استدعاء الإجراء settextbuf ذاكرة وسيطية قدرها 1024 بايت للملف textfile ويجب الانتباه إلى أن استدعاء الإجراء settextbuf بعد فتح الملف قد يفقدنا بعض المعلومات و كذلك يجب أن يكون المتحول الممثل للذاكرة الوسيطية متحولاً عاماً global variable لأن فقدان المتحول سوف يفقدنا جميع المعلومات المخزنة فيه و ستحدث أخطاء لم تكن في الحسبان.

تفريغ الذاكرة الوسيطية flushing buffer:

لقد ذكرنا سابقاً أن عملية الكتابة في الملفات تتم أولا بنقل المعلومات إلى الذاكرة الوسيطية buffer المخصصة لهذا الملف و التي تساوي كقيمة افتراضية 128 بايت و من ثم تنقل المعلومات إلى الملف الموجود على القرص في إحدى حالتين:

١. امتلاء الذاكرة الوسيطية و عدم مقدرتها على تخزين معلومات أخرى.
٢. إغلاق الملف.

ففي الحالتين السابقتين تُفرغ الذاكرة الوسيطية في الملف الموجود على القرص . لكن قد تحتاج أثناء سير البرنامج إلى تفريغ هذه الذاكرة الوسيطية عندما نريد نحن ذلك لذلك زودتنا لغة باسكال بالإجراء flush حيث ينقل هذا الإجراء المعلومات الموجودة في الذاكرة الوسيطية و يخزنها في القرص يستدعى هذا الإجراء عادة بعد استدعاء الإجراءات read وreadln و write و writeln ويأخذ استدعاء الإجراء flush متحولاً وسيطياً هو متحول الملف و المثال التالي يكتب المعلومات في الملف ويُفرغها على القرص مباشرة:

```
Write( filevariable,str);
Flush(str);
```

تجد الإشارة إلى أن عملية الوصول إلى محتويات الملفات النصية تتم بطريقة تسلسلية أي يجب قراءة السطر الأول قبل الثاني و الأسطر العشر الأولى قبل الحادي عشر و هكذا دواليك......و السبب في طريقة القراءة هذه أن أسطر الملف ليست بطول واحد لذلك لا يستطيع البرنامج معرفة السطر الحادي عشر مثلاً إلا من خلال قراءته للأسطر العشر الأولى قبل وصوله إلى السطر الحادي عشر.

الملفات محددة نوع المعطيات typed files:

تزودنا لغة تربو باسكال بنوعين من الملفات غير النصية هما : الملفات محددة نوع المعطيات typed files و الملفات مهملة (غير محددة) نوع المعطيات untyped files.

فالملفات محددة النوع عناصر تنتمي لنوع معطيات معين يملك حجماً محدداً.

يرينا البرنامج كيفية العمل مع الملفات محددة النوع:

code

```
program test;
const filename='c:\soso.int';
maxtrials=50;
type
Ifile=file of integer;
var samplefile:Ifile;
function ifmade(var thefile:Ifile; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
rewrite(thefile);
(*$I+*)
if IOResult=0 then
ifmade:=true
else
ifmade:=false
end;

function ifopened(var thefile:Ifile; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
reset(thefile);
(*$I+*)
if IOResult=0 then
ifopened:=true
else
ifopened:=false
end;
procedure dispints(value,precision,count,nrperline:integer);
begin
write(value:precision);

if count mod nrperline=0 then
writeln;
end;
procedure generateints(var thefile:Ifile;nr:integer);
```

```
var count,result:integer;
begin
for count:=1 to nr do
begin
result:=random(maxint);
write(thefile,result);
dispints(result,10,count,5);
end;
end;
procedure Freadints(var thefile:Ifile);
var count,result:integer;
begin
count:=0;

while not eof(thefile)do
begin
read(thefile,result);
inc(count);
dispints(result,10,count,5);
end;
end;
procedure drawline(length:integer; element:char);
var count:integer;
begin
for count:=1 to length do

write(element);

writeln;
end;
procedure wait;
begin
writeln('press enter to continue ');
readln;
end;
begin

randomize;
if ifmade(samplefile,filename) then
begin
generateints(samplefile,maxtrials);
close(samplefile);
drawline(50,'-');
if ifopened(samplefile,filename)then
begin
Freadints(samplefile);
close(samplefile);
end
else
writeln('could not open ',filename);
end
```

```
else
writeln('colud not cerate',filename);
wait;
end.
```

نفذ البرنامج السابق قبل الضغط على مفتاح الإدخال enter سوف ترى نسختين من 50 قيمة مفصولتين بسطر من النقاط.
إن البرنامج السابق يولد 50 قيمة صحيحة عشوائية و يخزنها في الملف و يظهرها على الشاشة و من ثم يقرأ هذه القيم من الملف و يظهرها مرة أخرى. إن الملف الذي أنشاه هذا البرنامج حجمه 100 بايت لأنه يحتوي على 50 قيمة صحيحة كل منها يحتاج 2bytes و هذا البرنامج سوف يعطينا خرج يشبه الخرج التالي:

```
11904    13832    31875    24085    15573
   16117    24671     6624    22583     5985
   15054    24083     1340    26593     3925
   14134    18778     3603    23840    22694
    6854      932    14888     7218    13907
   17396    26474    24390    18824    22113
    7243    29646    32462    17190    26384
   31395     6586    22759     8322     7589
   29051    18643     4264    16234    14192
   16237      677    22178     6092    16368
-------------------------------------------------
   11904    13832    31875    24085    15573
   16117    24671     6624    22583     5985
   15054    24083     1340    26593     3925
   14134    18778     3603    23840    22694
    6854      932    14888     7218    13907
   17396    26474    24390    18824    22113
    7243    29646    32462    17190    26384
   31395     6586    22759     8322     7589
   29051    18643     4264    16234    14192
   16237      677    22178     6092    16368
press enter to continue
```

بعد الانتهاء من تنفيذ البرنامج قم بفتح ملف soso.int بواسطة إحدى برامج محرر النصوص ستفاجئ برؤية رموز لا معنى لها على الشاشة و ذلك بسبب أن الملف ليس ملفاً نصياً لأن الملف قد خزن وفق الصيغة الثنائية binary format و ليس بصيغة ASCII (ASCII Format) لذلك لا نستطيع عرض هذه الملفات أو تنقيحها بواسطة محررات النصوص.

تعريف الملفات محددة نوع المعطيات defining a typed file:

إذا أردنا استخدام ملف عناصره من نوع معطيات عدد صحيح مثلاً عينا أولاً تعريف هذا النوع من الملفات أي يجب علينا تعريف نوع المعطيات الجد هذا قبل التصريح عن متحولات تنتمي لهذا النوع .
و الصيغة الكتابية لتعريف هذه الملفات محددة نوع المعطيات هي كالتالي:

<مميز>"=file of"<نوع معطيات أساسي>

نوع المعطيات الأساسي هذا يمك أن يكون أي نوع معطيات عدا نوع المعطيات ملف أو أي بنية معطيات data Structure تحتوي نوع المعطيات ملف.

نورد الآن بعض التعاريف الصحيحة لأنواع ملفات متعددة. سنضع ضمن التعليقات comments قبل التعريف حجم عناصر الملف بالبايت وسوف نستخدم مثل هذه التعاريف ضمن برامجنا:

(* 64 bytes:31 bytes per string+ 2 for integer);

```
Type person=record
First,last:string[30];
Age:integer;
End;
(* 40 bytes:4 bytes per element *)
Listate=array[1..10] of longint;

 (*60 bytes:6 bytes per elemenr*)
Rstats=array[1..10]of real;
(* 32 bytes:reguired to store 256 bits*)
Charset=set of char;
(* now define files containing elements of these types*)
Persfile=file of person;
LIFile=file of listats;
Rfile=file of Rstats;
Setfile=file of charset;
Ifile=file of integer;
```

إنشاء وفتح الملفات محددة النوع creating and opening typed files:

ينشئ الإجراء iFMade في البرنامج السابق ملفاً جديداً من النوع IFile أما الإجراء IFOpened فيفتح ملفاً موجود أصلاً وهذان الإجراءان مشابهان تماما للإجراءين السابقين Fmade وFopened المستخدمين لإنشاء وفتح الملفات النصية فالوسيط الأول لهذه الإجراءات عبارة عن متحول الملف والثاني اسم الملف .

و في الحقيقة تعامل الملفات نصية أم محددة لنوع بطريقة واحدة تقريباً فكما رأينا أن إنشاء وفتح الملفات محددة النوع مشابهة لإنشاء وفتح الملفات النصية وكذلك يستخدم الإجراء close لإغلاق الملفات محددة النوع وتخزين محتويات هذا الملف على القرص قبل إغلاقه . وقبل تطبيق أي عملية على الملفات محددة النوع يجب ربط متحول الملف باسم الملف لربط الملف بالعالم الخارجي للبرنامج أي القرص.

الكتابة على الملفات محددة النوع writing to typed files:

إذا نظرنا إلى البرنامج السابق نلاحظ أن البرنامج قد أنشأ ملف جديد من النوع IFile ومن ثم استدعي الإجراء Generateints الذي يولد الإعداد الصحيحة العشوائية random ومن ثم يكتب هذه القيم في الملف المحدد . وهذه القيم تكتب باستخدام الإجراء write حيث يملك استدعاء هذا الإجراء نفس الصيغة الكتابية لاستدعائه للكتابة على الملفات النصية حيث أن تربو باسكال سمحت باستخدام هذا الإجراء للكتابة على الملفات محددة النوع في حين إن الإجراء writeln غير مسموح به لأنه كما ذكرنا سابقاً يقوم بعد طباعة المعلومات الممررة إليه بإرسال اتحاد رمزي تحكم هما:رمز الإرجاع CR (carriage return) ورمز التغذية السطرية LF (linefeed) ويُرمز لاتحاد هذين الرمزين بالاختصار CRLF ووظيفة هذين الرمزين هي إرجاع المؤشر إلى بداية سطر جديد.

ولكن إذا كنا نتعامل مع ملفات محددة الأنواع فإن إرجاع المؤشر إلى بداية سطر جديد ليس له معنى في حين أنه مهم جداً إذ كنا نتعامل مع الملفات النصية .

إذاً من أجل الكتابة على الملفات النصية يستخدم الإجراء write فقط و لا يستخدم الإجراء writeln .

يكتب الإجراء write المعلومات في الموقع الحالي لمؤشر الملف و من ثم ينقل مؤشر الملف إلى الموقع التالي لانتظار أي معلومات جديدة.

القراءة من الملفات المحددة النوع reading from typed files:

يستدم الإجراء read من أجل قراءة المعلومات من الملف محدد النوع كما استخدم ضمن الملفات النصية. أما الإجراء readln فلا يُستخدم أبداً مع الملفات محددة النوع.

يقرأ الإجراءread المعلومات من الموقع الحالي لمؤشر الملف و من ثم ينقل مؤشر الملف إلى الموقع التالي لانتظار قراءة المعلومات التالية.

الوصول العشوائي للملفات محددة النوع random access to typed files:

تعتبر طريقة الوصول العشوائي إلى المعلومات إحدى الفوارق الأساسية التي تميز الملفات النصية عن الملفات محددة النوع حيث طريقة الوصول للمعلومات في الملفات النصية طريقة تسلسلية أي يجب قراءة المعلومات من بداية الملف وبالترتيب. أما الملفات محددة النوع فإن الأمر يختلف كثيراً فطول عناصر الملف محدد النوع معروف مسبقاً من خلال تعريف نوع الملف ولا توجد رموز لها معان خاصة ضمن الملفات محددة النوع مثل CRLF .لذلك يستطيع البرنامج

الوصول إلى أي موقع في الملف بشكل عشوائي وببساطة بواسطة حساب إزاحة offset هذا الموقع عن بداية الملف و من ثم التوجه إلى ذلك الموقع الذي يتضمن قيمة تنتمي لنوع المعطيات الأساسي للملف.
يمكننا الإجراء مسبق التعريف seek من الانتقال من أي موقع ضمن الملف إلى الموقع المطلوب ضمن الملفات محددة الأنواع و هذا الإجراء يأخذ متحولين وسيطين هما:

١. الملف محدد النوع المراد البحث ضمنه.
٢. عدد صحيح يحدد الموقع المراد الانتقال إليه ضمن الملف.

ترقم عناصر الملفات محددة النوع ابتداء من الرقم 0 فإذا كان لدينا ملف فيه 1000 عنصر فإن عناصر هذا الملف سوف ترقم ابتداءً من 0 و حتى 999 فمن أجل الانتقال إلى العنصر العاشر ضمن هذا الملف والذي موقعه position هو 9 سوف نستخدم العبارة التالية:

Seek(mytypedfile,9)

يرينا البرنامج التالي كيفية استخدام الإجراء seek حيث يكتب كمية كبيرة من الأعداد الحقيقية العشوائية حيث يكتب كمية كبيرة من الأعداد الحقيقية العشوائية في ملف ومن ثم يخزن هذا الملف. وبعد ذلك يعيد البرنامج فتحه لقراءة عناصر مختارة منه. حيث يتم تحديد هذه العناصر المختارة بواسطة تحديد موقعها و في مثالنا حددنا العناصر الأخيرة أي التي موقعها بين 1900 1999 .

code

```
program test;
const filename='c:\soso.int';
maxitems=2000;
type
Rfile=file of real;
var RF:Rfile;
count:integer;
whichelem:longint;
currval:real;
function rfmade(var thefile:rfile; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
rewrite(thefile);
(*$I+*)
if IOResult=0 then
rfmade:=true
else
rfmade:=false
end;

function rfopened(var thefile:rfile; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
reset(thefile);
(*$I+*)
if IOResult=0 then
rfopened:=true
else
rfopened:=false
end;
procedure dispints(value,precision,count,nrperline:integer);
begin
write(value:precision);
```

```
if count mod nrperline=0 then
writeln;
end;
procedure wait;
begin
writeln('press enter to contine ');
readln;
end;
begin{main}
randomize;
if rfmade(rf,filename)then
begin
for count:=1 to maxitems do
begin
currval:=random;
write(rf,currval);
end;
close(rf);
writeln('done');
if rfopened(rf,filename) then
begin
for count:=1 to 10 do
begin
whichelem:=random(100)+1900;
seek(rf,whichelem);
read(rf,currval);
writeln(whichelem:5,': ',currval:10:5);
end;
close(rf);
end
else
writeln('could not open ',filename);
end
else
writeln('colud not cerate',filename);
wait;
end.
```

خرج البرنامج شبيه بالخرج التالي:

```
done
 1937:   0.11046
 1994:   0.64324
 1977:   0.46193
 1923:   0.43171
 1957:   0.84239
 1965:   0.19722
 1992:   0.92935
 1989:   0.22642
 1928:   0.08828
 1917:   0.76406
```

لاحظ كيفية استخدام الإجراء seek الذي سوف يضع المؤشر عند العنصر المطلوب و هذا يعني أن القيمة التالية التي ستقرأ من الملف ستكون العنصر المطلوب نفسه.

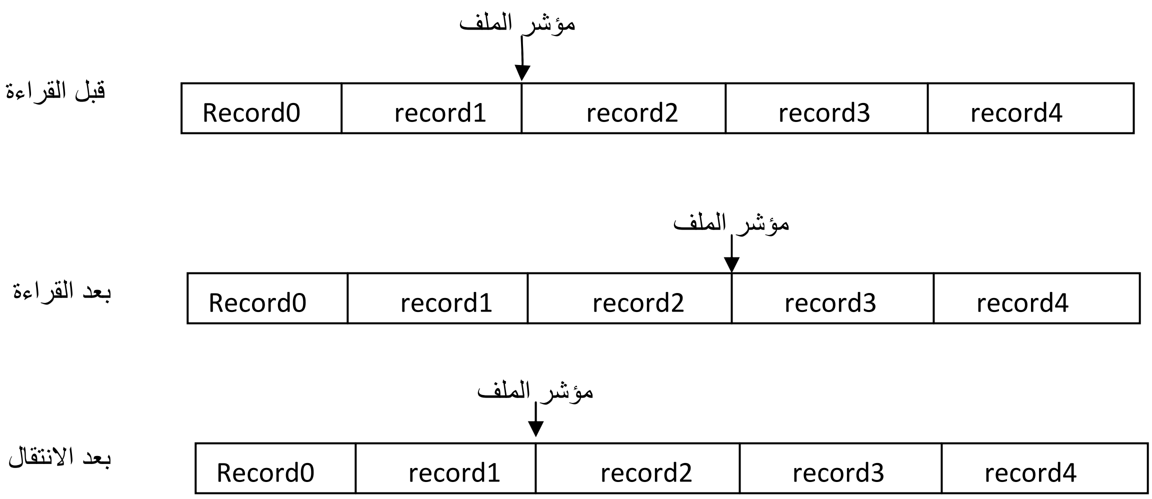
و من أجل إيضاح مفهوم مؤشر الملف file pointer و لتمثيل عملية انتقال المؤشر نورد التعريف التالي:

```
Type examplerecord=record
Field1, field2:integer;
End;
Examplefile=file of examplerecord;
Var efile:examplefile;
Value: examplerecord;
```

نلاحظ من خلال التعريف السابق أن نوع المعطيات examplefile قد عُرف على أنه نوع معطيات مؤلف من ملف محدد النوع و نوع معطياته الأساسي هو السجل examplerecord فإذا كان لدينا العبارات التالية:

```
Seek(efile,2);
Read(efile,value);
Seek(efile,2);
```

عندئذ يمكن تمثيل الملف السابق بالشكل التالي الذي يبين تنفيذ كل عبارة من العبارات السابقة:



إن طريقة الوصول العشوائية أسرع في الوصول إلى المعلومات المطلوبة منها من الطريقة التسلسلية و خاصة في الملفات الكبيرة. من أجل مقارنة سرعة هاتين الطريقتين لنفترض أن لدينا ملفاً نصياً يحتوي أيضاً على عدد كبير من القيم الحقيقية و نحن نريد قراءة العدد الموجود في الموقع الذي رقمه 1975 عندئذ سيقرأ البرنامج المعلومات السابقة لهذا الموقع كلها ابتداء من الموقع 0 و حتى الموقع 1974 ووصولاً إلى الموقع المطلوب و هو 1975 و من أجل ملاحظة الفرق هذا نفذ البرنامج التالي:

code

```
program test;
const filename='c:\soso.int';
maxitems=2000;
var RF:text;
count:integer;
whichelem,index:integer;
curval:real;
f:text;
function fmade(var thefile:text; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
rewrite(thefile);
```

```
(*$I+*)
if IOResult=0 then
fmade:=true
else
fmade:=false
end;

function fopened(var thefile:text; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
reset(thefile);
(*$I+*)
if IOResult=0 then
fopened:=true
else
fopened:=false
end;
procedure dispints(value,precision,count,nrperline:integer);
begin
write(value:precision);

if count mod nrperline=0 then
writeln;
end;
procedure
dispreals(value:real;width,precision,count,nrperline:integer);
begin
write(value:width:precision);
if (count mod nrperline=0)then
writeln;
end;
procedure wait;
begin
writeln('press enter to contine ');
readln;
end;
begin{main}
randomize;
if fmade(rf,filename)then
begin
for count:=1 to maxitems do
begin
currval:=random;
writeln(rf,currval:10:5);
end;
close(rf);
writeln('done');
for count:=1 to 10 do
begin
if fopened(rf,filename) then
```

```
begin
whichelem:=random(100)+1900;
for index:=1 to whichelem do
readln(rf,currval);
writeln(whichelem:5,': ',currval:10:5);
close(rf);
end
else
writeln('could not open ',filename);
end;
end
else
writeln('colud not cerate',filename);
wait;
end.
```

الخرج البرنامج شبيه بالخرج التالي:

```
done
 1925:   0.12637
 1960:   0.93992
 1947:   0.52022
 1967:   0.44525
 1933:   0.10704
 1988:   0.55615
 1941:   0.25577
 1962:   0.31712
 1905:   0.87877
 1959:   0.00949
```

إن أهم ما يلاحظ هنا أن الملفات النصية ينبغي إعادة فتحها في كل مرة نحتاج فيها لقراءة قيمة ما تقع قبل القيمة المقروءة لأن المؤشر الملف لا يعود تلقائياً إلى بداية الملف أو حتى ينقل تلقائياً إلى القيمة التالية .لذلك بعد قراءة القيمة اتي تقع في الموقع 1950 لا يمكننا قراءة القيمة التي تقع في الموقع 1949 بدون إعادة مؤشر الملف إلى بداية الملف أي علينا إغلاق الملف ومن ثم إعادة فتحه باستخدام الإجراء reset عندها سيعود مؤشر الملف إلى بداية الملف فمثلاً قلنا أن الوصول إلى العنصر 1950 يحتاج إلى المرور بكل العناصر السابقة له أي حتى العنصر 1949 ومن ثم نستطيع قراءة هذا العنصر.

أمثلة عن ملفات محددة النوع more typed-file examples:

١. ملف يحتوي على أنساق A File Containing Arrays:

يرينا البرنامج التالي كيفية تعريف ملف عناصره انساق تتألف من عشرين عنصراً ينتمي إلى نوع المعطيات عدد حقيقي. وهذا يعني أن كل عنصر من عناصر الملف سوف تأخذ من الذاكرة حجماً قدره 120 بايت أي 20×6bytes .

code

```
program test;
const filename='c:\soso.int';
maxarrays=3;
arrsize=20;
type rstats=array[1..arrsize] of real;
 RFile=file of rstats;
var rf:rfile;
rdata:rstats;
arrcount,count,nrelements:longint;
function rfmade(var thefile:rfile; fname:string):boolean;
```

```
begin
assign(thefile,fname);
{$I-}
rewrite(thefile);
(*$I+*)
if IOResult=0 then
rfmade:=true
else
rfmade:=false
end;

function rfopened(var thefile:rfile; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
reset(thefile);
(*$I+*)
if IOResult=0 then
rfopened:=true
else
rfopened:=false
end;

procedure drawline(length:integer;element:char);
var count:integer;
begin
for count:=1 to length do
write(element);
writeln;
end;
procedure
dispreals(value:real;width,precision,count,nrperline:integer);
begin
write(value:width:precision);
if (count mod nrperline=0)then
writeln;
end;
procedure wait;
begin
writeln('press enter to continue ');
readln;
end;
begin{main}
randomize;
writeln('rstats take ',sizeof(rstats),' bytes each');
if rfmade(rf,filename)then
begin
writeln('writing arrays ...');
for count:=1 to maxarrays do
begin
for arrcount:=1 to arrsize do
```

```
rdata[arrcount]:=random;
write(rf,rdata);
end;
close(rf);
writeln('done writing.');
if rfopened(rf,filename)then
begin
while not eof(rf) do
begin
drawline(50,'-');
read(rf,rdata);
for arrcount:=1 to arrsize do
dispreals(rdata[arrcount],10,5,arrcount,5);
end;
drawline(50,'-');
end
else
writeln('could not open ',filename);
nrelements:=filesize(rf);
writeln(filename,' contains ', nrelements,' elements(arrays) ');
close(rf);
wait;
end
else
writeln('could not create ',filename);
end.
```

إن خرج البرنامج يكون شبيهه بالخرج التالي:

```
rstats take 120 bytes each
done writing.
--------------------------------------------------
  0.31750  0.18063  0.79920  0.01810  0.05073
  0.69391  0.50642  0.78456  0.66682  0.85230
  0.48402  0.58049  0.08335  0.60682  0.05235
  0.58347  0.81795  0.44370  0.88355  0.10905
--------------------------------------------------
  0.97253  0.24347  0.98442  0.96250  0.17958
  0.94954  0.72713  0.02977  0.74467  0.21834
  0.78712  0.40468  0.46958  0.47272  0.09766
  0.59902  0.89060  0.03050  0.18035  0.63260
--------------------------------------------------
  0.91610  0.23022  0.46094  0.05003  0.09799
  0.72570  0.42997  0.16113  0.02129  0.37148
  0.75742  0.05188  0.54626  0.37098  0.72088
  0.16270  0.88745  0.60919  0.55654  0.48102
--------------------------------------------------
c:\soso.int contains 3 elements(arrays)
press enter to continue
```

نلاحظ من خلال البرنامج السابق أن استدعاء الإجراءين READ و WRITE لا يختلف عن استدعائهما من أجل ملف له نوع آخر يختلف عن نوع معطيات هذا الملف ولكن في الحقيقة تختلف كمية ونوع المعطيات التي يتعامل معها هذان الإجراءان من نوع إلى آخر و ذلك حسب نوع المعطيات الأساسي الخاص بالملف.
يحتوي البرنامج السابق على إجراء مسبق التعريف Filesize حيث يأخذ هذا التابع متحولاً وسيطياً واحداً عبارة عن متحول الملف محدد النوع و يعيد هذا التابع عدد العناصر المحتواة في هذا الملف و ليس عدد بايتات هذا الملف و في البرنامج السابق يحتوي الملف RF ثلاثة أنساق من النوع Rstats فإن هذا التابع سيعيد القيمة 3 في حين إن الملف يحتوي على 60 قيمة كل منها يحتاج إلى 6 بايت و بالتالي حجم الملف soso.int الكلي= 360 بايت.

٢. ملف يحتوي على سجل ومجموعة A File Containing Record and Set:

يمكننا إنشاء ملف محدد النوع أكثر تعقيداً حيث يستخدم البرنامج التالي نوع المعطيات سجل كنوع معطيات أساسي له و أحد حقول السجل تنتمي إلى نوع المعطيات مجموعة charset وهي عبارة عن مجموعة من الرموز .

code

```
program test;
const filename='c:\soso.int';
maxtrials=9+1;
type charset=set of char;
chrec=record
setinfo:charset;
maxitems:integer;
end;
chrecfile=file of chrec;
var chrf:chrecfile;
chrecdata:chrec;
setcount,nrtimes,arrcount,count:integer;
chindex:char;
function chrfmade(var thefile:chrecfile; fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
rewrite(thefile);
(*$I+*)
if IOResult=0 then
chrfmade:=true
else
chrfmade:=false
end;
function chrfopened(var thefile:chrecfile;
fname:string):boolean;
begin
assign(thefile,fname);
{$I-}
reset(thefile);
(*$I+*)
if IOResult=0 then
chrfopened:=true
else
chrfopened:=false
end;
procedure drawline(length:integer;element:char);
```

```
var count:integer;
begin
for count:=1 to length do
write(element);
writeln;
end;
procedure wait;
begin
writeln('press enter to continue ');
readln;
end;
begin {main program}
randomize;
writeln('charsets take ',sizeof(charset), 'bytes');
writeln('chrecs take ', sizeof(chrec),' bytes');
if chrfmade(chrf,filename) then
begin
for count:=1 to maxtrials do
begin
nrtimes:=random(27);
chrecdata.setinfo:=[];
chrecdata.maxitems:=nrtimes;
for setcount:=1 to nrtimes do
with chrecdata do
setinfo:=setinfo+[chr(random(26)+97)];{'a'..'z'}
write(chrf,chrecdata);
end;
close(chrf);
drawline(50,'-');
if chrfopened(chrf,filename) then
begin
while not eof (chrf) do
begin
read(chrf,chrecdata);
write('max = ',chrecdata.maxitems,' : ');
for chindex:='a' to 'z' do
begin
if chindex in chrecdata.setinfo then
write(chindex, ' ');
end;
writeln;
end;
drawline(50,'-');
writeln(filename,' contains ', filesize(chrf),' elements(chrecs) ');
wait;
end
else
writeln('could not open ',filename);
end
else
writeln('could not open ',filename);
```

```
end.
```

خرج البرنامج شبيهه بالخرج التالي:

```
charsets take 32bytes
chrecs take 34 bytes
--------------------------------------------------
max = 5 : f g n u y
max = 12 : a c f g h n y z
max = 10 : c j k l m n o r s
max = 12 : a d e f i l s v x
max = 10 : b e j m n x y
max = 24 : a c d e g h j l m n p q r w y z
max = 5 : a o p z
max = 23 : c d f g i j k l m p q r u v w x y z
max = 0 :
max = 9 : a f h j v x z
--------------------------------------------------
c:\soso.int contains 10 elements(chrecs)
press enter to continue
```

٣. ملف يحتوي على نوع معطيات تعدادي file containing enumerated type:

يمكننا تعريف ملف محتوياته تنتمي لنوع معطيات تعدادي و لكن قبل كل شيء علينا التذكر أننا لا نستطيع كتابة القيم التعدادية مباشرة على الشاشة أو على ملف نصي ولكن يمكننا كتابة هذه القيم في الملفات محددة النوع و ذلك لأن القيم الترتيبية يمكن تخزينها بسهولة أما إذا أردت إظهار هذه القيم على الشاشة فهناك عدة خطوات يجب إتباعها لإظهار القيم التعدادية.

والبرنامج التالي يبين لنا مثالاً عن ملف يحتوي نوع معطيات تعدادي:

code

```
program test;
const filename='c:\soso.int';
maxtrials=1+9;
type color=(red,green,blue,yellow,white,black);
cfile=file of color;
var
cf:cfile;
currcolor:color;
procedure wait;
begin
writeln('press Enter to continue.');
readln
end;
procedure drawline(length:integer;element:char);
var count:integer;
begin
for count:=1 to length do
write(element);
```

```
writeln;
end;
function cfmade(var thefile:cfile;fname:string):boolean;
begin
assign(thefile,filename);
{$I-*}
rewrite(thefile);
(*$I+*)
if IOResult=0 then
cfmade:=true
else
cfmade:=false
end;
function cfopened(var thefile:cfile;fname:string):boolean;
begin
assign(thefile,filename);
{$I-*}
reset(thefile);
(*$I+*)
if IOResult=0 then
cfopened:=true
else
cfopened:=false
end;
begin {main program}
if cfmade(cf,filename) then
begin
currcolor:=red;
write(cf,currcolor);
currcolor:=black;
write(cf,currcolor);
currcolor:=green;
write(cf,currcolor);
currcolor:=white;
write(cf,currcolor);
currcolor:=blue;
write(cf,currcolor);
currcolor:=yellow;
write(cf,currcolor);
close(cf);
if cfopened(cf,filename) then
begin
while not eof (cf) do
begin
read(cf,currcolor);
case currcolor of
red:writeln('red');
green:writeln('green');
blue:writeln('blue');
yellow:writeln('yellow');
white:writeln('white');
```

```
black:writeln('black');
end;
end;
writeln('number of elements = ',filesize(cf));
close(cf);

wait;

end
else
writeln('could not open ',filename);
end
else
writeln('could not create ',filename);
end.
```

خرج البرنامج هو:

```
red
black
green
white
blue
yellow
number of elements = 6
```

الملفات مهملة النوع untyped files:

تزودنا لغة تربو باسكال بملفات مهملة النوع untyped files (غير محددة النوع) ويصرح عنها بكلمة file كنوع معطيات و الملفات مهملة النوع عبارة عن مجموع من البايتات يجري نقلها على شكل دفعات بين البرنامج و الملف. وهذه تكون محددة الحجم ويتم تحديدها عند إنشاء الملف أو فتحه . فمن أجل كتابة المعلومات في الملف مهمل النوع يقوم البرنامج بنقل كمية من البايتات إلى الملف و كذلك الأمر إذا أردنا قراءة معلومات من ملف مهمل النوع و من ثم تفسر هذه المعلومات ضمن البرنامج.يستخدم الإجراءان blockread و blockwrite لكتابة المعلومات أو قراءتها من الملفات مهملة النوع أما الإجراءان READ و WRITE فلا يستخدمان مع هذا النوع من الملفات.
البرنامج القادم يشرح لنا كيفية استخدام الملفات مهملة النوع حيث يُولد البرنامج قيم عشوائية حقيقية و من ثم يكتبها وفق تتابع من البايتات في ملف مهمل النوع و كمية الملفات التي ستنقل بطول 6 بايت في كل عملية نقل و هذا يطابق عدد البايتات التي يحتاجها العدد الحقيقي. وبعد أن يُكتب الملف و يُغلق يقرأ البرنامج كتلاً من المعلومات من هذا الملف و يظهر القيم على الشاشة.

code

```
program test;
const filename='c:\soso.int';
maxtrials=1+9;
var ut:file;
count:integer;
val:real;
procedure wait;
begin
writeln('press Enter to continue.');
readln
end;
procedure drawline(length:integer;element:char);
var count:integer;
begin
```

```
for count:=1 to length do
write(element);
writeln;
end;
function utfmade(var
thefile:file;fname:string;chunksize:word):boolean;
begin
assign(thefile,filename);
{$I-*}
rewrite(thefile,chunksize);
(*$I+*)
if IOResult=0 then
utfmade:=true
else
utfmade:=false
end;
function utfopened(var
thefile:file;fname:string;chunksize:word):boolean;
begin
assign(thefile,filename);
{$I-*}
reset(thefile,chunksize);
(*$I+*)
if IOResult=0 then
utfopened:=true
else
utfopened:=false
end;
procedure dispreals(value:real;
width,precision,count,nrperline:integer);
begin
write(value:width:precision);
if count mod nrperline=0 then
writeln;
end;
begin {main program}
randomize;
if utfmade(ut,filename,6) then
begin
for count:=1 to maxtrials do
begin
val:=random;
blockwrite(ut,val,1);
dispreals(val,10,5,count,5);
end;
close(ut);
drawline(50,'-');
if utfopened(ut,filename,6)then
begin
while not eof(ut)do
begin
```

```
blockread(ut,val,1);
inc(count);
DISPREALS(val,10,5,count,5);
end;
drawline(50,'-');
wait;
close(ut);
end
else
writeln('could not open ',filename);
end
else
writeln('could not create ',filename);
end.
```

خرج البرنامج شبيه الخرج التالي:

```
  0.76653  0.76331  0.84420  0.87173  0.06428
  0.25005  0.28635  0.21733  0.93054  0.59206
--------------------------------------------------
  0.76653  0.76331  0.84420  0.87173  0.06428
  0.25005  0.28635  0.21733  0.93054  0.59206
--------------------------------------------------
press Enter to continue.
```

في البرنامج السابق نلاحظ أن حجم الملف الناتج soso.int حجمه 60 بايت.
ونلاحظ أن التابعين utfmade و utfopened فهذا الروتينان يأخذان وسيطاً ثالثاً يحدد عدد البايتات التي سوف تعتبر سجلا واحدا أو كما ذكرت سابقاً دفعة واحدة عند نقل المعلومات من و إلى الملف فمثلاً في البرنامج هذا الوسيط يأخذ القيمة 6 عند استدعاء هذا الإجراء ليتمكن من نقل عدد حقيقي كامل في كل عملية نقل بين البرنامج و الملف.
إن القيمة النظامية لحجم الدفعات التي تنقل بين البرنامج والملف هي 128 بايت لكن يمكننا اختيار حجم معين كما لاحظنا و هذا الحجم يعتمد على نوع المعلومات التي ستخزن ضمن الملف فإذا كنا نخزن قيماً صحيحة فيجب علينا نقل بايتين أو أربع بايتات في كل مرة أما إذا كان ملف مهمل النوع يخزن مزيجاً من المعلومات فان علينا استخدام خوارزمية آمنة في النقل و هي نقل المعلومات بايتا بايتا و من ثم يوكل للبرنامج مهمة تجميع هذه البايتات وربطها لبناء القيم التي نريد.
إذاً الوسيط chunksize يستخدم ضمن استدعاء الإجراءين لكي يتعاملا مع الملفات مهملة النوع ولا يستخدم عند التعامل مع الملفات محددة النوع أو الملفات النصية.
ولقد ذكرنا آنفاً أننا نستخدم الإجراء blockWrite من أجل كتابة المعلومات في الملفات مهملة النوع و هذا الإجراء يأخذ ثلاثة متحولات وسيطية و يمكن أن يأخذ متحولاً وسيطياً رابعاً إذا أردنا ذلك:

١. المتحول الوسيطي الأول هو متحول الملف مهمل النوع.
٢. المتحول الوسيطي الثاني يمكن أن يكون أي متحول أو موقع في الذاكرة في الذاكرة حيث يفهم هذا الإجراء هذا المتحول على أنه سلسلة مواقع من الذاكرة الوسيطية طولها يحدده المتحول الوسيطي الثالث و هذا المتحول الوسيطي الثاني يجب أن يمرر مرجعياً.
٣. المتحول الوسيطي الثالث يحدد عدد البايتات التي سوف تقرأ في كل مرة و هذا المتحول يجب أن يكون من النوع word و حجم هذه الدفعات كما ذكرت يعتمد على نوع المعطيات المستخدمة و هذا المتحول الوسيطي يجب أن يمرر بواسطة قيمته.
٤. يمكن إضافة متحول وسيطي رابع ينتمي أيضاً لنوع المعطيات word حيث يزودنا هذا المتحول الوسيطي بمعلومات حول عدد الدفعات التي كتبت بعد الانتهاء من تنفيذ هذا الإجراء ويعيد هذا الإجراء قيمة ضمن هذا المتحول الوسيطي تمثل عدد البايتات التي كتبها و هذا العدد يمكن أن بكون مساوياً لقيمة المتحول الوسيطي الثالث أو أقل و أيضاً يجب أن يمرر مرجعياً.

يملك الإجراء blockread نفس الوسطاء المستخدمة مع الإجراء blockwrite حيث يقرأ الإجراء Blockread عدداً محدداً من البايتات من ملف مهمل النوع و يخزن هذه البايتات في موقع من الذكرة محدد عبر وسيطه الثاني و الذي يمكن أن يكون متحولاً أيضاً و أخيراً حتى تعمل هذه الإجراءات يجب أن يفتح الملف مهمل النوع أولاً .

اعتبار الأجهزة كملفات Device As Files:

تنجز كل عمليات الإدخال و الإخراج في تربو باسكال عبر أجهزة كلوحة المفاتيح و الشاشة أو ملف ما على القرص و لتبسيط الأمر اعتبرت تربو باسكال كل الأجهزة ملفات مما سهل عملية برمجتها جداً.
تستخدم أجهزة النظام DOS لإنجاز كل عمليات الإدخال و الإخراج ومن هذه الأجهزة لوحة المفاتيح و الشاشة و الفأرة ووحدة أشرطة النسخ الاحتياطي و الأقراص الضوئية.
بالإضافة إلى بعض الأجهزة التي لا يدعمها النظام DOS و التي تتطلب برنامج تشغيل خاص بها device driver

أجهزة الدخل والخرج القياسي the standard input and output devices:

انطلاقاً من مفهوم أن عمليات الدخل و الخرج في تربو باسكال تتم عبر الإجهزة فإننا نستطيع القول بأن استدعاء الإجراء READLN يخبر تربو باسكال بان تقبل الإدخالات من جهاز الدخل القياسي وكذلك الأمر بالنسبة للإجراء WRITELN الذي يخبر تربو باسكال بأن ترسل المخرجات إلى جهاز الخرج القياسي.
يدعى الجهاز CON بجهاز الدخل والخرج القياسي console و هو يشير إلى لوحة المفاتيح كجهاز دخل و الشاشة كجهاز خرج. و على ذلك يمكن استخدام الجهاز CON كأي ملف من ملفات القرص و على ذلك يمكن استخدام الجهاز CON كأي ملف من ملفات القرص و بذلك يمكن استعماله مع إجراءات مثل reset و rewrite كما في البرنامج التالي:

code

```
program test;
var
textfile:text; str:string;
begin
assign(textfile,'con');
rewrite(textfile);
writeln(textfile,'output to con');
writeln;
writeln(textfile,'enter string using readln(str)');
write('type a string press enter when done. ');
readln(str);
writeln(textfile,'>',str);
assign(textfile,'con');
reset(textfile);
writeln('enter string from using readln(textfile,str)');
write('type a string press enter when done. ');
readln(textfile,str);
writeln('>',str);
writeln;
close(textfile);
write('press enter for exit ');
readln;
end.
readln;
end.
```

أجهزة الطابعة printer devices:

يدعم النظام Dos عدة أجهزة طابعة هي : PRN و LPT1 و LPT2 و LPT3 علماً بأن الجهاز PRN هو نفسه LPT1. يستخدم معظم الناس طابعة واحدة موصولة مع منفذ الطابعة الأول لذلك يشار عادة إلى الطابعة بالجهاز PRN. تستخدم الطابعة كجهاز خرج فقط لذلك إذا حاولنا فتح الجهاز PRN للقراءة عبر الإجراء RESET فإن تربو باسكال سوف تولد نهاية للملف مباشرة(أي سيعيد التابع EOF القيمة TRUE).
إذاً لإرسال المعلومات إلى الطابعة يُفتح ملف الجهاز PRN للكتابة و تجري عليه عملية الكتابة كما تعلمنا.
توجد طريقة أخرى في الكتابة على الطابعة و ذلك عبر استخدام وحدة مسبقة التعريف في تربو باسكال هي printer .تحتوي هذه الوحدة على متحول ملف نصي مسبق التعريف يدعى LST وظيفته توجيه المخرجات مباشرة إلى الطابعة.

مثال توضيحي على ذلك:

```
Program test;
Uses printer;
Begin
Writeln(LST,'syria arabic');
Readln
End.
```

الأجهزة التسلسلية serial device:

تملك معظم الأجهزة منافذ تسلسلية serial ports تستخدم لربط الطابعات أيضاً و الموديمات و الشبكات المحلية و لأغراض اتصالات أخرى.
تدعم لغة تربو باسكال جهازين تسلسلين هما com1 و com2 بالإضافة إلى الجهاز الوسيطي Aux و الذي يشير إلى com1 و بالتالي فخما متكافئان.
و مع أن هذه الأجهزة جعلت إرسال و استقبال المعلومات عبر المنافذ التسلسلية أمراً سهلاً إلا أن استخدامها محدود لأغراض خاصة جداً.

الجهاز Nul the NUL Device:

تملك لغة تربو باسكال جهاز هو الجهاز NUL و لهذا الجهاز استخدام خاص مميز جداً إذ أنه يتجاهل أي شيْ يُرسل إليه. و لابد أنك تستغرب أهمية هكذا جهاز حيث أن أهميته تكمن عند تصميم برنامج اتصالات يقوم بإرسالات فعندما نريد تجريب مراحل البرنامج عدا مرحلة الإرسال فإن الجهاز عندها سيحل محل الجهاز الذي نريد الإرسال إليه . وبعد الانتهاء من اختبار مراحل البرنامج التي لا تتعلق بالإرسال يمكن إعادة الجهاز افصلي و تجريب مرحلة الإرسال فقط.

أمثلة:

السؤال الأول:

لدينا ملف نصي f1.txt مكتوب بالغة الإنكليزية يراد طباعة معلومات عن معلومات موجودة ضمن الملف على الشكل التالي:

```
number line          count words in line          long words in line
----------------     -------------------------    -----------------------

Count all words                count all line
-------------------            -----------------
```

حيث number line تمثل رقم كل سطر من أسطر الملف و count words in line تمثل عدد الكلمات الكلية في السطر و long words in line تمثل أطول كلمة في السطر(الكلمة التي تحوي أكثر عدد من الأحرف).
Count all words تمثل عدد الكلمات الكلية في الملف و count all line تمثل عدد الأسطر الكلية في الملف.
حيث يفصل بين الكلمة و الأخرى فراغ واحد على الأقل.
حيث يطلب كتابة إجرائية print_file لطباعة معلومات عن الملف بالشكل الموصف أعلاه.
كتابة إجرائية print لطباعة الملف كما هو مخزن على القرص.

Code

```
PROGRAM TEST;
procedure print_file;
var allwords,count,max,ln:integer;
c:char; s:string;  f:text;
begin
assign(f,'c:\f1.txt');
reset(f);
```

```
allwords:=0;
ln:=0;
writeln('number line         count words in line          long words in line');
writeln('-----------          -------------------          -------------------');
while not eof(f) do
begin
ln:=ln+1;
count:=0;
max:=0;
s:='';
write(' ',ln:3);
while(not eoln(f))do
begin
read(f,c);
if(c<>' ')then
s:=s+c;
if(c=' ')and(s<>'') or eoln(f) then
begin
count:=count+1;
if(length(s)>max)then
max:=length(s);
s:='';
end
end;
writeln(' ':28,count:3,' ':28,max);
allwords:=allwords+count;
readln(f);
end;
close(f);
writeln('count all words            count all line');
writeln('---------------            ---------------');
writeln(' ',allwords,' ':30,ln);
end;
procedure print;
var toto:text; c:char;
begin
assign(toto,'c:\f1.txt');
reset(toto);
while(not eof(toto))do
begin
read(toto,c);
write(c);
end;
writeln;
close(toto);
end;
begin
print;
writeln('----------------------------------------');
print_file;
readln
```

```
end.
```

لنفرض أن محتويات الملف f1.txt كالشكل التالي:

C:\F1.txt
khaled yassin alsheikh syria arabic welcome to syria arabic good luck syriaaaa   welcome

يكون خرج البرنامج:

```
khaled yassin alsheikh
syria arabic
welcome to syria arabic
good luck
syriaaaa        welcome
number line            count words in line           long words in line
-----------            -------------------           -------------------
   1                           3                              8
   2                           2                              6
   3                           4                              7
   4                           2                              4
   5                           2                              8
count all words              count all line
---------------              ---------------
 13                                5
```

السؤال الثاني:

لدينا الملف ثلاث ملفات نصية f1,f2,f3 يراد طباعة هذه الملفات الثلاث على شاشة الحاسب على الشكل التالي:

the first line in file1	the first line in file2	the first line in file3
the second line in file2	the second line in file2	the second line in file3
the end of file1	this is long line>>>	the end of file3
	another line from file2	

مع مراعاة النقاط التالية:

- يطبع سطر الملف الأول ثم يليه سطر الملف الثاني بعد 3 فراغات ثم يليه سطر الملف الثالث بعد 3 فراغات .
- في حال تجاوز طول سطر الملف عن 20 محرف يتم إهمال باقي الحروف و يضاف إلى السطر المحارف التالية <<< للدلالة على ذلك.
- في حال انتهاء ملف قبل الآخر يتم الاستعاضة عن كل سطر من اسطر الملف المنتهي بمجموعة من الفراغات.

Code

```
program test;
procedure print3file;
var f1,f2,f3:text; c1,i:integer;
c:char;
begin
assign(f1,'c:\f1.txt');
assign(f2,'c:\f2.txt');
assign(f3,'c:\f3.txt');
reset(f1);reset(f2);reset(f3);
while(not eof(f1) or not eof(f2) or not eof(f3))do
begin
c1:=0;
i:=0;
while(not eoln(f1))do
```

```
begin
i:=i+1;
read(f1,c);
if(i>20)then
begin
write('>>>');
break;
end;
write(c);
c1:=c1+1;
end;
if(i>20)then
for i:=1 to 3 do
write(' ')
else
for i:=c1 to 25 do
write(' ');
i:=0; c1:=0;
while(not eoln(f2))do
begin
i:=i+1;
read(f2,c);
if(i>20)then
begin
write('>>>');
break;
end;
write(c);
c1:=c1+1;
end;
if(i>20)then
for i:=1 to 3 do
write(' ')
else
for i:=c1 to 25 do
write(' ');
i:=0; c1:=0;
while(not eoln(f3))do
begin
i:=i+1;
read(f3,c);
if(i>20)then
begin
write('>>>');
break;
end;
write(c);
c1:=c1+1;
end;
readln(f1);
readln(f2);
```

```
readln(f3);
writeln;
end;
close(f1);
close(f2);
close(f3);
end;
begin
print3file;
readln;
end.
```

بفرض محتويات الملفات النصية على لشكل التالي:

C:\f1.txt	c:\f2.txt	c:\f3.txt
khaled yassin alsheikh syria arabic welcome to syria Arabic good luck syriaaaa welcome	khaled yassin al sheikh welcome Syria Arabic in Damascus Arabic Syria	good luck good bye see you tomorrow

خرج البرنامج هو :

```
khaled yassin alshei>>>   khaled yassin alshei>>>   good luck
syria arabic              Syria Arabic in Dama>>>   good bye see you tom>>>
welcome to syria Ara>>>   syria
good luck
syriaaaa        welco>>>
```

عدل البرنامج السابق بحيث لا يتم احتساب الفراغ كمحرف.

الحل التعديل بسيط هو أن نضع فقط

```
If(c<>'  ')then
i:=i+1;
```

Code

```
program test;
procedure print3file;
var f1,f2,f3:text; c1,i:integer;
c:char;
begin
assign(f1,'c:\f1.txt');
assign(f2,'c:\f2.txt');
assign(f3,'c:\f3.txt');
reset(f1);reset(f2);reset(f3);
while(not eof(f1) or not eof(f2) or not eof(f3))do
begin
c1:=0;
i:=0;
while(not eoln(f1))do
begin
read(f1,c);
```

```
if c<>' 'then
i:=i+1;
if(i>20)then
begin
write('>>>');
break;
end;
write(c);
c1:=c1+1;
end;
if(i>20)then
for i:=1 to 3 do
write(' ')
else
for i:=c1 to 25 do
write(' ');
i:=0; c1:=0;
while(not eoln(f2))do
begin
read(f2,c);
if(c<>' ')then
i:=i+1;
if(i>20)then
begin
write('>>>');
break;
end;
write(c);
c1:=c1+1;
end;
if(i>20)then
for i:=1 to 3 do
write(' ')
else
for i:=c1 to 25 do
write(' ');
i:=0; c1:=0;
while(not eoln(f3))do
begin

read(f3,c);
if(c<>' ')then
i:=i+1;
if(i>20)then
begin
write('>>>');
break;
end;
write(c);
c1:=c1+1;
end;
```

```
readln(f1);
readln(f2);
readln(f3);
writeln;
end;
close(f1);
close(f2);
close(f3);
end;
begin
print3file;
readln;
end.
```

خرج البرنامج بعد التعديل هو :

```
khaled yassin alsheikh     khaled yassin al sheikh>>>    good luck
syria arabic               Syria Arabic in Damascu>>>    good bye see you tomorro>>>
welcome to syria Arabic    syria
good luck
syriaaaa       welcome
```

السؤال الثالث:

اكتب برنامج بلغة باسكال القياسية يقوم بدمج ملف نصي f2 إلى ملف نصي f1

Code

```
program test;
procedure merge;
var f1,f2:text;
var c:char;
begin
assign(f1,'c:\f1.txt');
append(f1);
assign(f2,'c:\f2.txt');
reset(f2);
while(not eof(f2))do
begin
read(f2,c);
write(f1,c);
end;
close(f2);
close(f1);
end;
begin
merge;
readln
end.
```

اكتب برنامج بلغة باسكال لحذف الكلمات ذات الطول الزوجي في ملف نصي f1.txt بحيث يراد إعادة تشكيل هذا الملف بحيث لا يحوي الكلمات ذات الطول الزوجي.

Code

```
program test;
procedure remove_length_even;
var f1,f2:text; c:char; s:string;
begin
assign(f1,'c:\f1.txt');
reset(f1);
assign(f2,'c:\f2.txt');
rewrite(f2);
while(not eof(f1))do
begin
s:='';
while(not eoln(f1))do
begin
read(f1,c);
if(c<>' ')then
s:=s+c;
if((c=' ')and(s<>'')) or eoln(f1) then
begin
if length(s) mod 2=1 then
write(f2,s,' ');
s:='';
end
{else
if(c=' ')then
write(f2,c);  }
end;
writeln(f2);
readln(f1);
end;
close(f1); close(f2);
end;
procedure replay;
var f1,f2:text; c:char;ok:boolean;
begin
assign(f1,'c:\f1.txt');
rewrite(f1);
assign(f2,'c:\f2.txt');
reset(f2);
while(not eof(f2))do
begin
ok:=false;
while(not eoln(f2))do
begin
ok:=true;
read(f2,c);
write(f1,c);
write(c);
end;
```

```
if( ok)then
writeln(f1);
writeln;
readln(f2);
end;
close(f1); close(f2);
erase(f2);
end;
begin
remove_length_even;
replay;
readln;
end.
```

بفرض محتويات الملف f1.txt كالتالي:

```
C:\f1.txt
khaled yassin alsheikh
syria arabic
welcome to syria Arabic
good luck
syriaaaa      welcome
```

تصبح محتويات الملف f1.txt كالتالي:

```
C:\f1.txt
syria
welcome syria
welcome
```

اكتب إجراء replace_word لاستبدال كلمة arabic أينما وردت ضمن ملف نصي f1.txt بكلمة syria

```
Code
program test;
procedure replace_word;
var f1,f2:text; c:char; s:string;
begin
assign(f1,'c:\f1.txt');
assign(f2,'c:\f2.txt');
reset(f1);
rewrite(f2);
while not eof(f1) do
begin
s:='';
while not eoln(f1) do
begin
read(f1,c);
if c<>' 'then
s:=s+c;
if(c=' ')and (s<>'') or  eoln(f1) then
begin
if(s='arabic')then
```

```
begin
write(f2,'syria',' ');
s:='';
end
else
if s<>'' then
begin
write(f2,s,' ');
s:='';
end;
end
else
if(c=' ')then
write(f2,c);
end;
writeln(f2);
readln(f1);
end;
close(f1);
close(f2);
end;
procedure replay;
var f1,f2:text; c:char;
begin
assign(f1,'c:\f1.txt');
rewrite(f1);
assign(f2,'c:\f2.txt');
reset(f2);
while(not eof(f2))do
begin
read(f2,c);
write(f1,c);
end;
close(f1);close(f2);
erase(f2);
end;
begin
replace_word;
replay;
readln
end.
```

بفرض محتويات الملف النصي c:\f1.txt هي:

F1.txt
welcome to syria arabic welcome   to   syria arabic arabic in damasucs arabic arabic arabic arabic in Damascus arabic

يصبح محتويات الملف بعد التنفيذ هو:

F1.TXT
welcome to syria syria welcome to syria syria syria in damasucs syria syria syria syria in Damascus syria

اكتب برنامج بلغة تربو باسكال يقوم بحذف الأسطر الحاوية على كلمة hi في ملف نصي f.txt.

Code

```
program test;
var g:set of 1..255;
procedure replace_word;
var f1,f2:text; c:char;
s:string;
i:integer;
procedure replay;
var f1,f2:text; c:char;
ok:boolean;
begin
i:=1;
assign(f1,'c:\f1.txt');
reset(f1);
assign(f2,'c:\f2.txt');
rewrite(f2);
while(not eof(f1))do
begin
if  not (i in g  ) then
begin
while not eoln(f1) do
begin
read(f1,c);
write(f2,c);
end;
end;
i:=i+1;
readln(f1);
writeln(f2);
end;
close(f1);
close(f2);
rewrite(f1);
reset(f2);
while(not eof(f2))do
begin
ok:=false;
while(not eoln(f2))do
begin
ok:=true;
```

```
read(f2,c);
write(f1,c);
end;
if(ok)then
writeln(f1);
readln(f2);
end;
close(f1);close(f2);
end;
begin
assign(f1,'c:\f1.txt');
assign(f2,'c:\f2.txt');
reset(f1);
rewrite(f2);
i:=1;
g:=[];
while not eof(f1) do
begin
s:='';
while not eoln(f1) do
begin
read(f1,c);
if c<>' 'then
s:=s+c;
if((c=' ')and (s<>'')) or eoln(f1) then
if s='hi' then
begin
g:=g+[i];
s:='';
break;
end
else
s:='';
end;
i:=i+1;
readln(f1);
end;
close(f1);
replay;
end;
begin
replace_word;
readln
end.
```

مثال بفرض محتويات الملف النصي f1.txt

```
F1.txt
hi to syria arabic
welcome     to     syria  arabic hi
arabic in damasucs
arabic arabic arabic arabic in Damascus
arabic hi
```

تصبح محتويات الملف بعد التنفيذ:

F1.txt
arabic in damasucs arabic arabic arabic arabic in Damascus

البرنامج السابق فعال من أجل عدد أسطر الملف =255 سطر كحد أقصى .
و المطلوب عدل البرنامج السابق ليصبح فعال من أجل ملف يحوي ألاف الأسطر.

اكتب برنامج بلغة turbo Pascal يقوم بإعادة تشكيل ملف نصي f1.txt بحيث يتم إعادة تشكيل (بناء) هذا الملف بحيث ينتهي كل سطر من أسطر الملف بعدد يعبر عن عدد الكلمات في السطر. علما أن الفاصل بين الكلمة هو فراغ واحد على الأقل ' '.

Code

```
program test;
procedure mov;
var f1,f2:text;
count:integer;
s:string;
c:char;
ok:boolean;
begin
assign(f1,'c:\f1.txt');
reset(f1);
assign(f2,'c:\f2.txt');
rewrite(f2);
while(not eof(f1))do
begin
s:='';
ok:=false;
count:=0;
while(not eoln(f1))do
begin
ok:=true;
read(f1,c);
if c<>' ' then
s:=s+c;
if(c=' ')and (s<>'')or eoln(f1)then
begin
count:=count+1;
write(f2,s,' ');
s:='';
end
else
if c= ' ' then
```

```
write(f2,c);
end;
if(ok)then
writeln(f2,count)
else
writeln(f2);
readln(f1);
end;
close(f1); close(f2);
end;
procedure replay;
var f1,f2:text;
c:char;
begin
assign(f1,'c:\f1.txt');
assign(f2,'c:\f2.txt');
reset(f2); rewrite(f1);
while(not eof(f2))do
begin
while(not eoln(f2))do
begin
read(f2,c);
write(f1,c);
end;
writeln(f1);
readln(f2);
end;
close(f1); close(f2);
erase(f2);
end;
begin
mov;
replay;
readln;
end.
```

بفرض محتويات الملف f1.txt كالتالي:

F1.txt
hi to syria arabic welcome   to   syria arabic hi arabic in damasucs arabic arabic arabic arabic in Damascus arabic hi syria arabic syria khaled yassin alsheikh syria

يصبح محتوى الملف f1.txt كالتالي:

```
hi to syria arabic 4
welcome      to      syria  arabic hi 5
arabic in damasucs 3
arabic arabic arabic arabic in Damascus 6
arabic hi 2
syria arabic syria 3
khaled yassin alsheikh 3
syria 1
```

دمشق- معضمية الشام 30/5/2011.

اللهم صلي و سلم وبارك على سيدنا محمد و على آله وصحبه أجمعين

محمد اشرف الأعراب و العجم　　محمد خير من يمشي على قدم
محمد باسط المعروف جامعه　　محمد صاحب الإحسان و الكرم
محمد ذكره روح لأنفسنا　　محمد شكره فرض على الأمم

اللهم لا تحسبنا بأعمالنا (السيئة)

يقول الإمام الشافعي رحمه الله تعالى:

أحفظ لسانك أيها الإنسانُ　　لا يلدغنك أنه ثعبانُ
كم في المقابر من قتيل لسانه　　كانت تهاب لقاءه الشجعانُ
جراحات السنان لها التام　　ولا يلتام ما جرح اللسانُ

(وما توفيقي إلا بالله الغفور الرحيم)